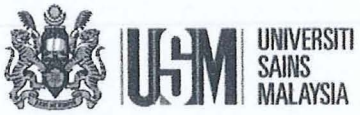
TECHNICAL REPORT

SHORT TERM GRANT
NO. 304/PKOMP/638137

A STUDY ON TECHNIQUES FOR HANDLING TRANSMISSION ERROR OF IPV6 PACKETS OVER FIBER OPTIC LINKS

PROF. MADYA DR. RAHMAT BUDIARTO
PROF. DR. SURESWARAN RAMADASS

December 2010
School of Computer Sciences
Universiti Sains Malaysia



Contents

1. Frame work of the research

Problem identification of the existing error control mechanism is very important to find out a new suitable design to solve the problem of ineffective error control. The identification results become main basic of designing a new mechanism. Hence, the design obtained truly solves the problem accurately. This section gives an overview of the proposed error control design that attempts to reduce delay on IPv6 packets transmission due to duplicate CRC verification and regeneration in router. Before presenting a design of a new error control mechanism, this section summarizes the problem identification and related works of the research. The summary is represented as a schematic diagram shown in Figure 1.

Figure 1 is a schematic diagram that represents summary of theoretical explanation including drawbacks of the existing error control mechanism, related works proposed to solve the drawbacks, advantages of features of IPv6 and HSN, and current requirements of data transfer. Each of them is put in the different envelope with arrow that shows relation one another. The following sub section will present analysis of each envelope to obtain particular rationales of the optimization method. Hence, it can support obtaining a new error control mechanism.

There are two limitations of current IPv6 packets transmission which are overhead and duplicate CRC calculation. The highest overhead is caused by processing overhead such as checksum computation and error checking. This is because the two processes touch each byte of the message that requires much time to do. Checksum computation is done in Transport layer as an error control field in the layer which is checksum field. While CRC calculation is conducted in Data Link layer of all nodes in a network. CRC calculation is one part of error control process in lower layer. Hence, the two limitations of IPv6 packets transmission mostly due to the error control mechanisms.

The weaknesses of current error control mechanism introduced bottleneck of IPv6 packets transmission over high speed networks. The important features of IPv6 including simpler header format, supporting for mobility, extensibility are not used optimally. This Section investigates the strength of the new protocol which is IPv6 and medium used in high speed networks. This is very important in order to obtain an

appropriate mechanism to reduce the bottleneck utilizing current feature of the existing technology. A novel error control mechanism can be created by combining the two advance technologies. Thus it has minimum side effect to the current system and reduces packet processing time and overhead.

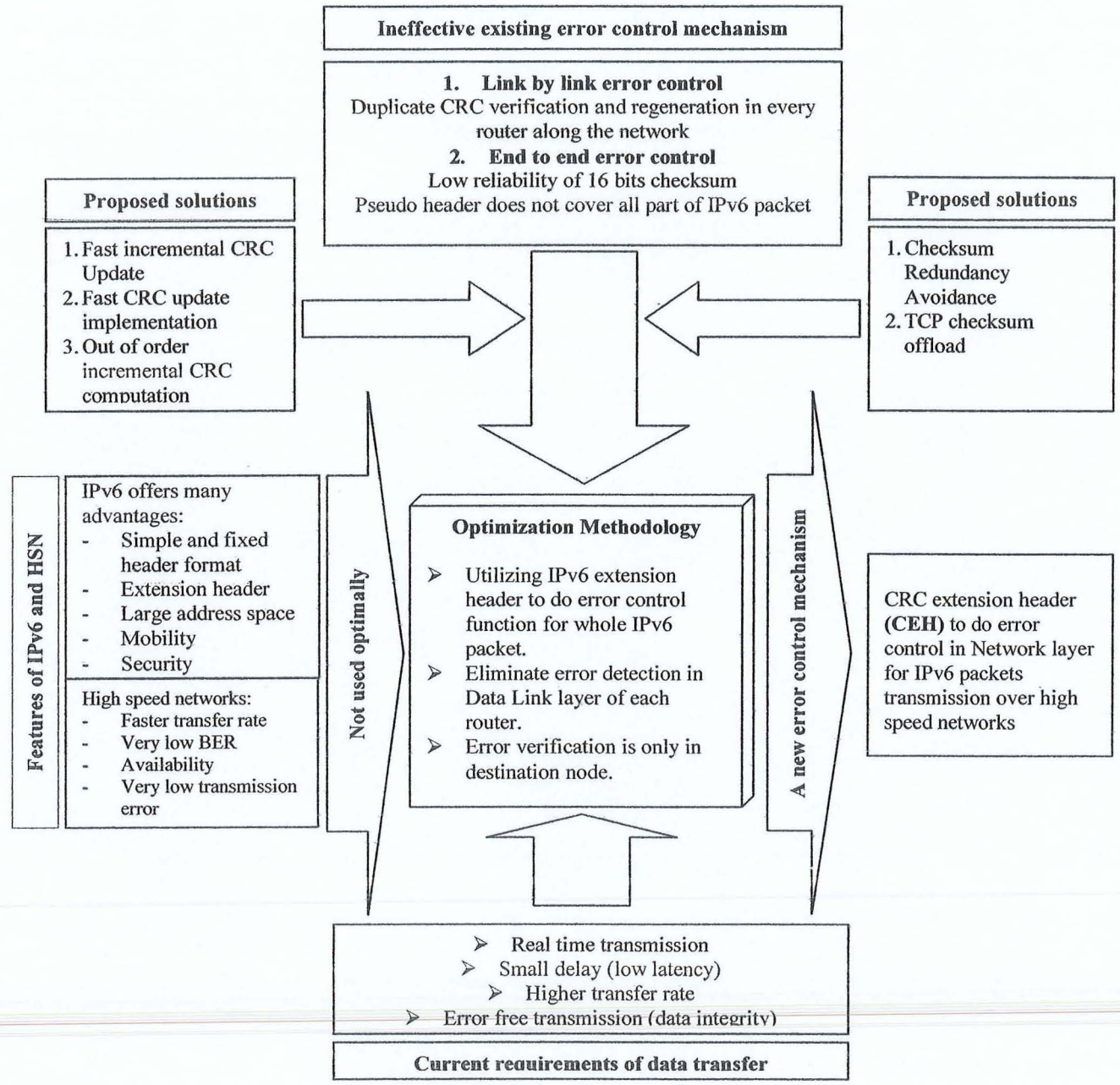


Figure 1 Summary of Theoretical Frameworks

2. Proposed model of error detection at Network layer

This research proposes a new IPv6 extension header. General extension header is placed after destination address field before upper layer data as shown in Figure 2.

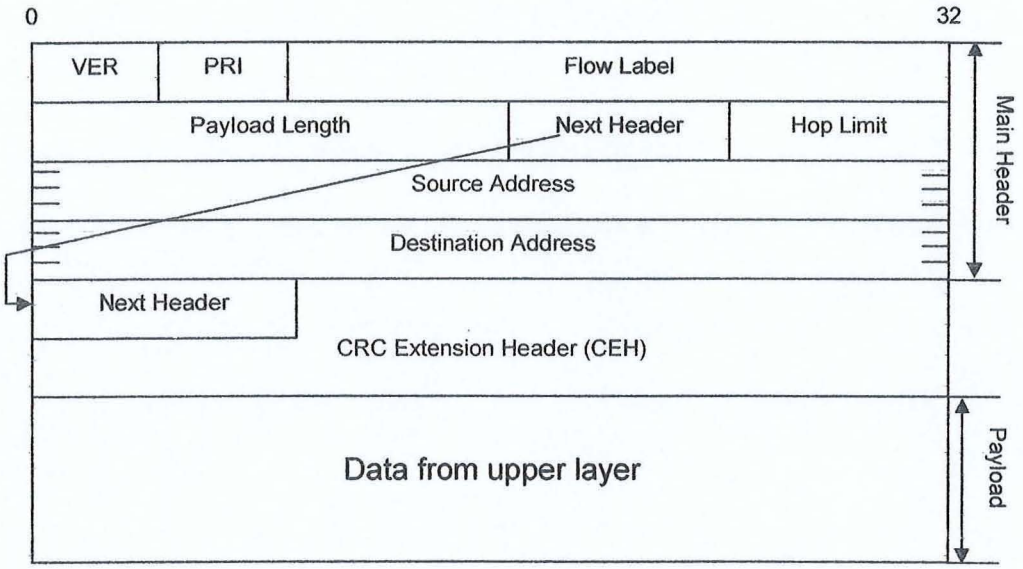


Figure 2 Structure of IPv6 with CRC Extension Header

The new IPv6 extension header is named CRC Extension Header (CEH). It has three fields as depicted in Figure 3.

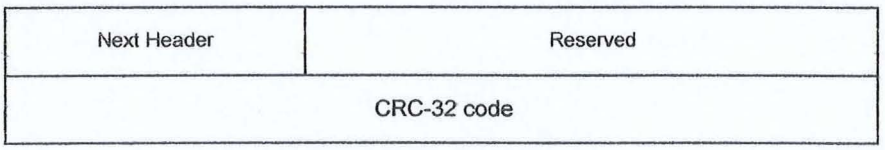


Figure 3 Format CRC Extension Header

Next header: 8 bits selector to identify the type of extension header or payload immediately following this extension header. This field has the value as allocated by IANA as listed in Table 2.2 such as 6 for TCP and 4 for IPv4. This field is mandatory of all extension headers as chain connection in order to speed up packet processing.

Reserved: this field is allocated for future improvement of CEH and is set to zero by the sender. RFC 2460 recommended extension header as multiple of 8 bytes. Thus, reserved field is padded to meet the minimum size of extension header.

CRC-32 code: this 32 bits is the main field of CEH that contain CRC code generated from the whole IPv6 packet excluding hop limit. The code will be verified by receiver once the IPv6 packet reach particular destination. This field will examine whether the packet contain error or not.

Method of error detection at Network layer is shown in Figure 4. Sender generates IPv6 packets in the Network layer and then generates CEH from the correspond packet excluding 1 byte hop limit. The packet delivers to Data Link layer to get Data Link header without trailer (FCS). The packets are then sent through interconnecting devices (routers) to reach the receiver. All the routers process the packet as usual excluding verification and regeneration of CRC code. They also do not process the CEH inside IPv6 packet. The receiver receives the IPv6 packets transmitted and verifies the CEH in its Network layer. When the receiver detects error in the packet received, it has to discard the packet and wait for retransmission.

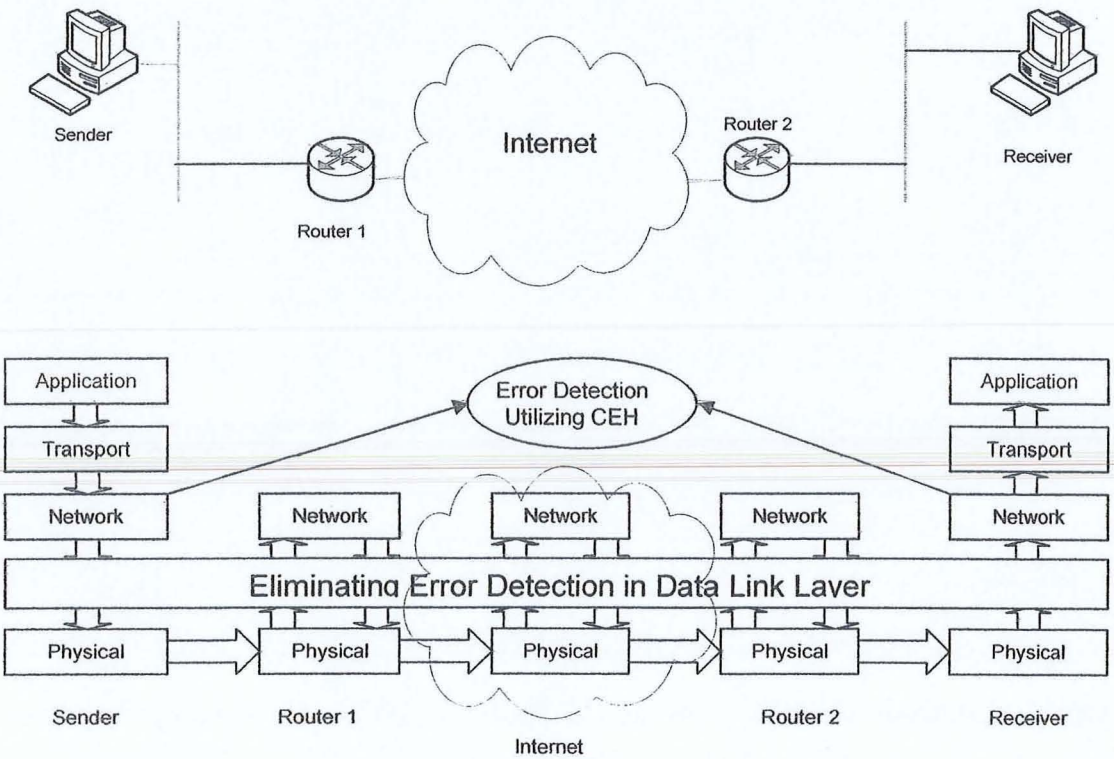


Figure 4 Method of Error Detection at Network Layer

This idea is extremely different with the existing method that conducts error detection and correction in Data Link layer. This new mechanism does not require the Data Link layer to verify and regenerate CRC code for error detection. Verification of CRC code will be done at Network layer of the destination node by processing CEH inside the IPv6 packet received. Routers just need to process IPv6 packet at their Network layer as usual which is simply a forwarding decision.

3. Experimental Setup

The main focus of this experiment is to transmit IPv6 packet either with CEH or with FCS in network topology in Figure 5. The aims are to measure the performance of IPv6 packet transmission with CEH as the error control. Thus, the experiments do not consider various routing protocols but instead uses IPv6 static routing. The configuration was done on Windows XP environment. IPv6 address for each on the experiment is listed in Table 1.

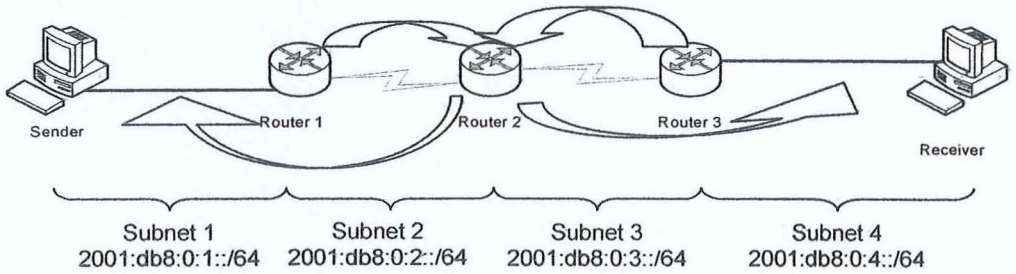


Figure 5 Experimental Setup

Table 1 IPv6 Address for All Nodes in the Experiment

No.	Role	MAC Address	IPv6 Address
1.	Sender	00:21:70:FD:E4:0E	2001:0DB8:0000:0001:0221:70FF:FEFD:E40E
2.	Router 1 (R1)	00:60:97:D2:5D:E4	2001:0DB8:0000:0001:0260:97FF:FED2:5DE4
		00:19:21:3B:F5:94	2001:0DB8:0000:0002:0219:21FF:FE3B:F594
3.	Router 2 (R2)	00:19:D1:22:1B:12	2001:0DB8:0000:0002:0219:D1FF:FE22:1B12
		00:04:75:ED:AA:F6	2001:0DB8:0000:0003:0204:75FF:FFED:AAF6

4.	Router 3 (R3)	00:10:5A:87:77:B4	2001:0DB8:0000:0003:0219:21FF:FE3b:FB19
		00:19:21:3B:FB:19	2001:0DB8:0000:0004:0210:5AFF:FE87:77B4
5.	Receiver	00:19:D1:22:1B:12	2001:0DB8:0000:0004:0219:21FF:FE3C:3B5E

4. Result

The total processing time of the experimental network of IPv6 packet with CEH transmission is lesser than IPv6 packet with FCS. Comparison of network latency (one way delay) on IPv6 packets transmission on the experimental network is listed in Table 2. Transmission of IPv6 packet with CEH is 68 % lower than transmission with FCS as error control mechanism. The decrement of network latency on transmission of IPv6 packet with CEH is because of elimination of error detection process in intermediate system. The process at intermediate node of IPv6 packet transmission with FCS actually has higher processing time than packet processing either at sender or receiver. Thus, eliminating the verification and regeneration in every router process extremely decrease overall processing time.

Table 2 Percentage of Decreasing Network Latency

No.	Packet Size (bytes)	L_{FCS} (ms)	L_{CEH} (ms)	ΔL (ms)	Percentage
1.	64	4.476	1.257	3.220	72%
2.	128	4.527	1.285	3.242	72%
3.	256	4.635	1.355	3.280	71%
4.	512	4.806	1.483	3.323	69%
5.	1024	4.926	1.681	3.245	66%
6.	1280	5.111	1.730	3.381	66%
7.	1500	5.334	1.804	3.530	66%

5. Conclusion

There are two achievements of this research. Firstly, structure of CRC extension header (CEH) proposed in Figure 3 has demonstrated better performance. The CEH is a new extension header that is not available before. Even though CEH mechanism is similar with destination option extension header, it has specific function on IPv6 packets transmission which is error detection. Secondly, the design of new error control mechanism with CEH depicted in Figure 4 showed error detection process utilizing CEH at Network layer is only done in sender and receiver. There is no error detection

process in Data Link layer of intermediate node. The result showed transmission of IPv6 packet with CEH as error control method in Network layer has decreased network latency of 68% average.

APPENDICES

A. GANTT CHART

B. PUBLICATIONS

C. FINANCIAL REPORT

Gantt chart of Research
Handling Transmission Error for IPv6 Packets over Fiber Optic Links

No	Activity	2008								2009												2010				
		5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5
1.	Literature Review																									
2.	Modeling																									
3.	Set up experiment requirements																									
4.	Data Collecting																									
5.	Data Analyze																									
6.	Presentation the findings																									
7.	Write up the report																									
8.	Submit report							6						12						18						24
9.	Final Report																									

Publication List

1. Supriyanto, Raja Kumar Murugesan, Rahmat Budiarto, Sureswaran Ramadass, *Handling Transmission Error for IPv6 Packets over High Speed Networks*, Proceedings of the 4th International Conference on Distributed Framework for Multimedia Applications (DFMA 2008), Penang, 21-22 Oct., 2008, pp. 159-163.
2. Supriyanto, Abidah M. Taib, Rahmat Budiarto. *Selecting a Cyclic Redundancy Check (CRC) Generator Polynomial for CEH (CRC Extension Header)*, Proceedings of Internasional Conference on Quality in Research, Jakarta. 3 – 6 August 2009, ISSN 114-1284 pp. 245 – 250.
3. Supriyanto, Raja Kumar Murugesan, Rahmat Budiarto, Sureswaran Ramadass, *CRC Extension Header (CEH): A New Model to Handle Transmission Error for IPv6 Packets over Fiber Optic Links*, Proceedings of IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009), October 4-6, 2009, Kuala Lumpur, Malaysia.
4. Supriyanto, Iznan H. Hasbullah, Rahmat Budiarto, *Exploiting IPv6 Extension Header to Handle Transmission Error for IPv6 Packets Over High Speed Networks*, Proceedings of International Conference on Advanced Computer Science and Information System, 7 – 8 December 2009, Jakarta, Indonesia
5. Bassam Altamimi, Abidah Hj Mat Taib, Rahmat Budiarto, *Protecting Teredo Clients from Source Routing Exploits*, Proceedings of the 4th International Conference on Distributed Framework for Multimedia Applications (DFMA 2008), Penang, 21-22 Oct., 2008, pp. 126-133.
6. Supriyanto, Manjur Kolhar, Rahmat Budiarto, Zaenal H, *Error Detection using CRC Extension Header for IPv6 Packets Transmission over High Speed Networks*, Europeon Journal of Scientific Research, (EJSR), ISSN 1450-216X Vol.43 No.2 (2010), pp.192-203 (SCOPUS Indexed).

Handling Transmission Error for IPv6 Packets over High Speed Networks

Supriyanto Praptodiyono, Raja Kumar Murugesan, Rahmat Budiarto, Sureswaran Ramadass

National Advanced IPv6 Centre of Excellence, Universiti Sains Malaysia, Penang

11800 USM, Penang, Malaysia

{supriyanto, raja, rahmat, sures}@nav6.org

***Abstract*— The escalating growth of web based services has led to the rapid growth of the Internet. As such the current Internet Protocol version 4 (IPv4) is rapidly running out of IP addresses. A new generation of IP known as IPv6 proposed and developed by the Internet Engineering Task Force (IETF) is gaining popularity in wide-spread use today. IPv6 offers many advantages including large address space, simple header, security, mobility and extensibility, over IPv4. In the Internet, transmission of data from one computer to another passes through a series of layers with control information added as headers at each layer. The header overheads can be reduced if the functionalities in the headers can be handled efficiently. In this paper we propose to remove error handling at the Data Link layer and handle it at the Network layer by utilizing the features of the IPv6 header especially extension header, and the characteristics of the high speed network medium. The proposed concept would reduce the overhead of the header at the Data Link layer resulting in increasing the data transfer rate and reducing the bandwidth utilization of IPv6 packets over high speed networks.**

***Keywords*— FCS, Cycle Redundancy Check, IPv6, TCP/IP model.**

I. INTRODUCTION

Computer Networks follow a layered architecture to reduce the design complexity [1]. Each layer is defined based on its preceding layer and has a set of well defined functions with clear cut boundaries. Also with layered architecture the implementation details of each layer is independent of other layers. When data are sent from one machine to the other, they pass through a series of layers before it reaches the other side. At each layer the packet is encapsulated with a header that contains control information to handle the data received at the other side by the corresponding layer. These headers may be an overhead in terms of processing and bandwidth utilization. If the header length can be reduced by handling efficiently the redundant functions we can reduce the packet handling or processing time and the bandwidth being used. This can be achieved by exploiting the characteristics or capabilities offered by the communication medium used to transfer data, and by improving the existing mechanisms handling data at the layers.

The current widely used Internet Communication Protocol in the world is IPv4 (Internet protocol version 4). Owing to its limitations in terms of depleting address space, and other demands arising out of the rapid explosion of the use of the Internet, mobile and handheld devices, and other services a Next Generation Protocol called IPv6 was developed.

The main improvement brought by IPv6 is the increase in the number of addresses available for networked devices. IPv4 supports 2^{32} (about 4.3 billion) addresses, which is inadequate for giving even one address to every living person, let alone supporting embedded and portable devices. IPv6, however, supports 2^{128} addresses or approximately 5×10^{28} addresses for each of the roughly 6.5 billion people alive today [2]. Likewise, the recent growth of high speed networks in terms of link layer technology namely, Gigabit Ethernet facilitates high speed data transfer in the order of Gigabits per second.

In computer networks, errors normally occur at the communication medium called transmission errors caused by the medium due to interference such as White or Gaussian noise and Cross talk [3]. The most common approaches to detect the errors are parity check and cyclic redundancy check (CRC) techniques [4]. The errors that are detected by either parity or cyclic redundancy check can be corrected with two types of mechanism called Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC) [5].

Usually the error check mechanism is applied in Data Link layer using Frame Check Sequence field. This paper attempts to remove error detection from the Data Link layer and handle it at the Network layer utilizing the capabilities and features available with IPv6 header and the advantages of high speed network medium. This paper also intends to foster further discussion in this area.

II. PACKET TRANSMISSION

The two popular network architectures that have been widely used are the ISO-OSI model and the TCP/IP model. While the ISO-OSI model has remained still popular as a reference model for its simplicity and clarity of functions, the TCP/IP was a

978-1-4244-2312-5/08/$25.00 ©2008 IEEE

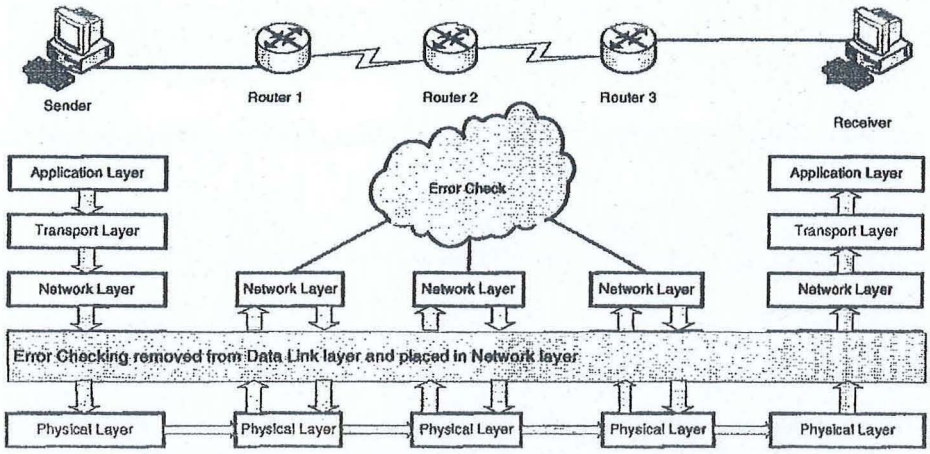


Fig. 4 The New Error Check Model

III. METHODOLOGY

In this paper, we intend to place the error detection function in the Network Layer to check the whole IPv6 packet. The error check mechanism used will be the same CRC method that is currently being used with the existing system. We will be experimenting to place the error detection in the extension header. The error check mechanism will compute the checksum for the whole packet in each hop. Fig. 4 shows the proposed model for handling errors in the Network layer.

In the new model there is no error checking in the Data Link Layer. We use high speed network as the communication link to transmit data as it is faster and more reliable. If there are errors detected by CRC on a hop or at the receiver side, the error will be corrected by requesting for a retransmission of the packet sent. This mechanism is called ARQ (Automatic Repeat reQuest).

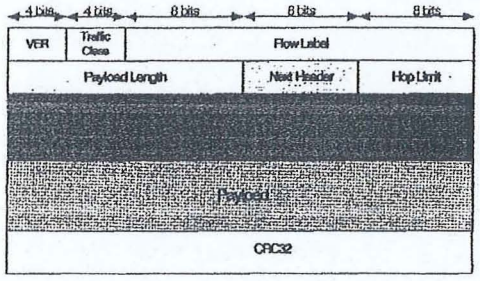


Fig. 5 New IPv6 Packet with CRC32

Fig. 5 shows the new format of IPv6 packet with CRC 32 placed in the extension header. This packet is generated on the sender side and sent to the receiver by copy of the CRC 32 code.

We will implement the proposed mechanism of error check placed in the Network layer and compare it with the existing mechanism of error check placed in the Data link layer. The comparison will help us to study the performance of the proposed mechanism in terms of error checking, percentage of packet loss, bandwidth usage and latency.

IV. DISCUSSION

Tere Parnell [8] states that Data Link layer encodes and frames data for transmission, in addition to providing error detection and flow control. Since the Data Link layer performs error checking, the same services need not be handled by the next higher layer. However, when a reliable medium is used, there is performance advantage by not handling error control in the Data Link layer, and instead handle it in another higher layer.

In terms of layering concept error control in general is handled at two different levels i.e. error control for upper layers and error control for lower layers. For the first, error control is handled by Transport layer. For IPv6 both TCP and UDP should apply checksum in its header. While for the lower layers error control is handled by the Data Link layer. The Data link layer uses CRC32 to check for transmission errors caused by the transmission medium.

As both Data link layer and Network layer are lower layers where the protocol is between neighboring nodes we can place the error check mechanism in the Network layer instead of the Data link layer. With today's technology and faster routers this can be accomplished without compromising the data transfer rate. On the hosts it would also be efficient to do error checking at the packet level instead of at the frame level.

A. Cyclic Redundancy Check

Cyclic redundancy check or CRC is the most widely used method of error detection on computer

V. CONCLUSION

In this paper we have proposed a new concept or method to move error handling from the Data Link layer to Network layer by utilizing the capabilities and features of the IPv6 protocol and the characteristics of high speed networks communication medium.

The proposed method reduces the overhead in terms of header length at the Data Link layer by removing the CRC field from its frame header and placing it in the extension headers of the IPv6 packet header in the Network layer. This proposed concept would enhance the performance of packet transmission in terms of reduced bandwidth utilization and faster data transfer rate, and thus enhance the performance of packet transmission.

ACKNOWLEGMENT

This work was supported by the short term grant for the project "A Study on Techniques for Handling Transmission Error for IPv6 Packets over Fiber Optic Links" 304/PKOMP/638137 funded by Universiti Sains Malaysia.

REFERENCES

[1] Tanenbaum, A.S., *Computer Networks*, Fourth Edition ed., Prentice Hall of India. New Delhi, 2006.

[2] Davies, J., *Understanding IPv6*, Microsoft Press, Washington: Microsoft Press, 2003.

[3] Mir, N.F., *Computer and Communication Networks*, Pearson Education Inc., Crawfordsville, Indiana, 2006.

[4] Shukla, S. and N.W. Bergmann. *Single bit error correction implementation in CRC-16 on FPGA*. In Proceedings of the IEEE International conference on Field-Programmable Technology, pp. 319 -322, 2004.

[5] Demir, U. and O. Aktas. *Raptor versus Reed Solomon forward error correction codes*. In the International Symposium on Computer Networks, pp. 264 - 269, 2006.

[6] Forouzan, B.A., *Data Communications and Networking*, Fourth Edition ed., Tata McGraw-Hill Publishing Company Limited, New Delhi India, 2006.

[7] Deering, S. and R. Hinden, *Internet Protocol Version 6 (IPv6) Specification*, in Request for Comments: 2460, The Internet Society, December 1998.

[8] Parnell, T., *Building High-Speed Networks* 1st edition, Mcgraw-Hill, 1999.

[9] Koopman, P. *32-bit cyclic redundancy codes for Internet applications*. In Proceedings of the International Conference on Dependable Systems and Networks, 2002.

[10] Koopman, P. *Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks*. In Proceedings of the International Conference on Dependable Systems and Networks, 2004.

[11] Henriksson, T. and L. Dake, *Implementation of fast CRC calculation*. In Proceedings of the Design Automation Conference, Asia and South Pacific, ASP-DAC, pp. 563 - 564, 2003.

[12] Chen, S.-y., and Y.-b. Li, *Error Correcting Cyclic Redundancy Checks based on Confidence Declaration*. In Proceedings of the 6th International Conference on ITS Telecommunications, pp. 511 - 514, 2006.

Selecting a Cyclic Redundancy Check (CRC) Generator Polynomial for CEH (CRC Extension Header)

Supriyanto[a], Abidah M. Taib[b], Rahmat Budiarto[c]

[a,c]*School of Computer Sciences*
Universiti Sains Malaysia, Penang Malaysia 11800
[a]*Email: supriyanto@nav6.org*
[c]*Email: rahmat@cs.usm.my*

[b]*Department of Computer Science*
Universiti Teknologi Mara (UiTM) Perlis Malaysia
Email: abidah@perlis.uitm.edu.my

ABSTRACT

Computation and regeneration of CRC code in each router may cause slower IPv6 packet transmission. Utilizing advantages of IPv6 features namely IPv6 extension header and fiber optic medium, we proposed CRC extension header (CEH) to do error control in Network layer rather than in Data Link layer. The purpose is to reduce error checking process in IPv6 packet transmission over high speed networks. The CEH will utilize CRC-32 to do error detection. This paper investigates which CRC-32 generator polynomial would be suitable for CEH. To find out the answer we developed a simulation program in Java that generates IPv6 packet and CRC-32 code. The simulation produced CRC processing time both at the sender and receiver. The result showed that CRC-32 generator polynomial proposed by Castagnoli is the fastest generator polynomial to generate CRC code. We then conclude that Castagnoli generator (CRC-32C) is the best generator to apply in CEH on IPv6 transmission over high speed networks.

Keywords

CRC-32, error, generator polynomial, transmission

1. INTRODUCTION

Internet has been connecting million people in the world. They use Internet not only to communicate each other but also military, business and administration purposes. Sending information from one side to another via Internet is fast and easy but may get some changes on the data due to weakness of the medium or noise affecting the channel. To ensure the data is accurate and free from error, designers have already equipped the protocol stacks such as TCP/IP with error control mechanism.

Error control in TCP/IP is divided into two types: error control in upper layer and lower layer. Upper layer is Transport layer that employ TCP or UDP checksum for segment data. While lower layer is Data Link layer that handle transmission error. This paper focuses on lower layer error control that ensures link by link data transfer of adjacent node is free from error. Transmission errors which change one or more bit of data may be caused by medium used in data transmission. Following the OSI reference model, data from Network layer will be added with header and trailer at the Data Link layer. Trailer is actually frame check sequence (FCS) that contains cyclic redundancy check (CRC) 32 bits to do error checking for the whole fields of link layer frame.

The traditional protocol stacks such as TCP/IP does error checking process by calculating and regenerating the CRC code in each intermediate node. It has to calculate CRC-32 of each IPv6 packet in incoming port of router and regenerate CRC-32 code before forwarding to the next hop. In response to increasing network speed and advance of fiber optic technology, the error checking mechanism in each router eventually become a bottleneck [1]. This study addresses improving IPv6 packet processing by eliminating CRC computation in each router. We proposed a new IPv6 extension header called CRC Extension Header (CEH) to do error checking in Network layer. The CEH uses the same CRC-32 algorithm which is table lookup algorithm. Format of CEH can be seen in figure 1 [2] below.

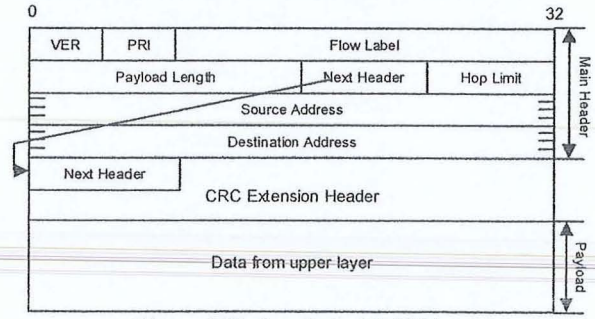


Figure 1 IPv6 packet with CEH

To generate CRC-32 code, it needs certain generator polynomial g(x). There are many generator polynomial

have been proposing by researchers. This paper intends selecting the best generator polynomial to be used in CEH. The rest of this paper is structured as follows. Section 2 describes the work by explaining overview of CRC operation. Section 3 discuses the candidates of generator polynomial that will be used in CEH. In Section 4, method of the selection is presented followed with result and discussion in section 5. The end of this paper is conclusion.

2. OVERVIEW OF CYCLIC REDUNDANCY CHECK

Cyclic redundancy check (CRC) is a code that used to detect errors that occur during transmission or storage of digital data information. Error detection is conducted by adding a code on the data transmitted and check the code at the receiver. There are various lengths of CRC codes such as 16 bits (CRC-16) [3] and 32 bits (CRC-32) [4]. This paper focuses on CRC-32 to be used in CEH to detect transmission error in Network layer. This section gives overview of CRC algorithm.

There are many algorithms utilized to generate CRC-32 code. This section introduces the simplest algorithm in order to understand the concept easily. Furthermore, it discusses the fastest algorithm which is table lookup algorithm that widely used today. The simplest way to know CRC code generation is algebraic approach as discussed in [5] and more explanation in [6]. We present a short overview of the algorithm.

In the algebraic approach, an information or message is interpreted as coefficient of polynomial called data word d(x). The data word is divided by pre determined CRC generator polynomial g(x) using equation 1 giving a code word c(x).

$$c(x) = q(x).g(x) + r(x) \quad (1)$$

The operation called modulo two division, meaning that all of the term on equation 1 is base two. q(x) is quotient of the division and r(x) is remainder of the division. CRC operation concerns on the remainder rather than quotient q(x). The remainder also could be obtained using equation 2, which m is the number of bits of the generator polynomial g(x) used or the highest degree of its polynomial. Data word d(x) is appended by 0s and represents with multiplication x^m.

$$r(x) = d(x).x^m \bmod g(x) \quad (2)$$

This operation to obtain r(x) is performed at the transmitter side. Code word c(x) which is d(x) + r(x) is transmitted along the network to reach a destination. Let say the receiver gets a code word c'(x) from the sender. Receiver conducts similar operation using the same generator.

$$c'(x) = c(x) + e(x) \quad (3)$$

There are two ways to justify whether there is an error in the code word received. Firstly, divide c'(x) using the same generator, if c'(x) is divisible by g(x) meaning there is no error or the error is undetected. Secondly, extract r(x) from the code word. Do operation of equation 2 using the same g(x) to get a new CRC code (remainder) r'(x) and then compare the r'(x) obtained with original r(x) extracted from the code word. If the two remainder is the same, there is no transmission error otherwise there is an error in the packet received.

Modulo 2 division generally could be performed by a sequence of shifts register. The division process makes addition and subtraction equal to bitwise XOR. Bitwise XORs are performed for all data including 0s appending till finish and obtain the remainder. The process is equal to shift bit by bit data to register from left most bits. In the hardware implementation uses LFSR (linier feedback shift register) of length m.

The process of bit by bit register needs more time and make it inefficient. To address the problem, engineers implement CRC operation in software using table lookup algorithm. The algorithm processes byte by byte instead of bit. The algorithm proposed by Sarwate [7]. *Firstly*, the CRC value is set to an initial value. Then data word is inputted to data stream byte by byte. *Secondly*, every byte of input stream is performed an XOR operation with the least significant byte of initial value. The result is used as index to access a 256 entry table. *Thirdly*, the value from the table is XORed with the rest of initial value by shifted byte by byte to the right. The result is CRC value to the next iteration.

In term of Ethernet, data word is the whole frame except frame check sequence field (FCS). As shown in figure 2, a standard Ethernet frame consist of destination and source Ethernet address, Ethernet type, data from upper layer and FCS.

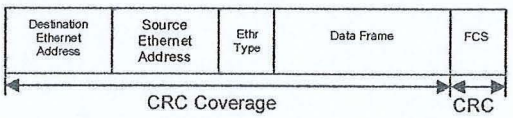


Figure 2 Standard Ethernet frame format

CRC coverage area in figure 2 is data word and FCS is remainder of modulo two division in equation 1. Hence, the whole frame is code word that actually transmitted into the networks.

3. IMPLEMENTATION CRC-32 FOR CEH

Implementation of CRC-32 in CEH uses the same algorithm with the existing CRC in Data Link layer which is table lookup algorithm. The differences of the new mechanism are coverage area and field of the CRC-

32. As mentioned previously, the CRC-32 is used as IPv6 extension header thus it placed between IPv6 main header and upper layer data *(see figure 1)*. The coverage area of CEH is the whole IPv6 packet excluding hop limit and CEH itself as shown in figure 3. From the figure, the size of CEH coverage area consist of 40 bytes IPv6 header minus 2 bytes hop limit field and upper layer data with maximum value 65,535 bytes. However the implementation is adapted with the maximum transmission unit (MTU) of the widely used Data Link layer technology which is Ethernet. Hence, the maximum MTU is 1500 bytes.

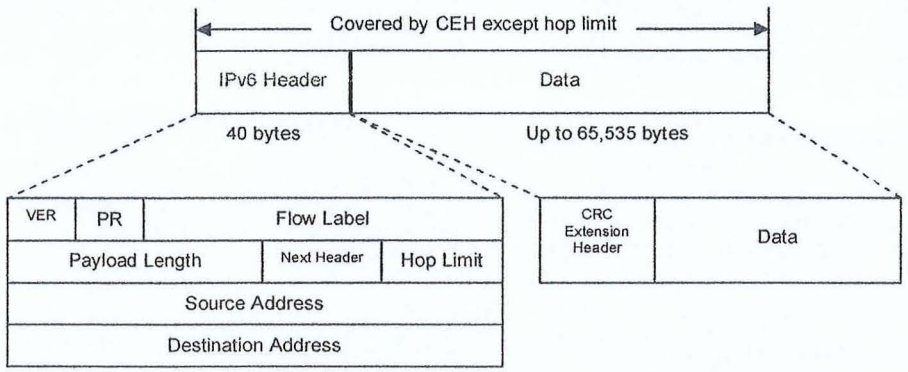


Figure 3 CEH coverage areas for error detection

Using equation 2, the coverage area is divided by g(x) to obtain r(x). r(x) is remainder of the division and represents of CRC code. Then, it is placed in the extension header field as shown in figure 4. Accordingly, pass the IPv6 packet to Data Link layer as usual to transmit through Physical layer without generate link layer trailer.

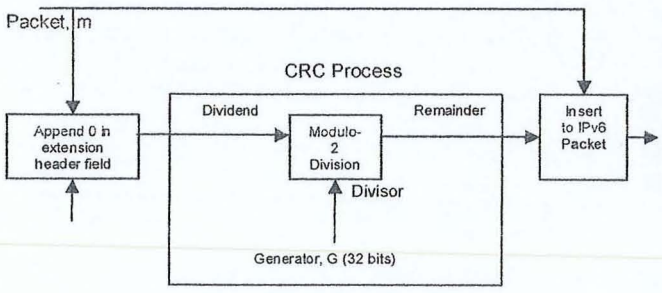


Figure 4 Generation of CEH in transmitter

At the receiver, Data Link layer captures the transmitted packet and pass the packet into Network layer directly. The layer does not compute CRC code any more. In the Network layer of the final destination, the packet will be processed as figure 5. It extracts CEH field from IPv6 packet. The other parts of the packet are divided by a generator to get a new CRC code. The new CRC code is compared with the original CRC inside CEH. If the result is similar, meaning no error. Thus, forward the packet to Transport layer, otherwise reject the packet.

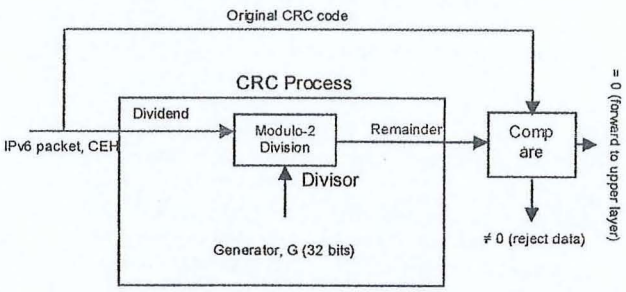


Figure 5 CEH processing in receiver

4. CANDIDATES OF GENERATOR POLYNOMIAL FOR CEH

Generator polynomial is the most important part of CRC code generation. It influences to the result of error detection. Hence selection of the best generator polynomial is also very important. This section discuses three candidates of generator polynomial that will be used in CEH they are standardized generator in IEEE 802.3 (CRC-32E), generator suggested by Guy Castagnoli (CRC-32C) and generator introduced by Philip Koopman (CRC-32K).

4.1 Generator used in Ethernet (CRC-32E)

This generator is widely used in data communication and data storage today. It was standardized by project IEEE 802.3 [8]. It is a polynomial with highest degree 32 and shown as

$$g(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+ x^{5}+x^{4}+x^{2}+x+1 \quad (4)$$

This polynomial can be formed as binary number as 100000100110000010001110110110111. The left most

1 is coefficient of x^{32} and following bits correspond to coefficient of the polynomial. The number of bits in a generator polynomial is equal to m + 1. The generator can be represented as 32 bit hexadecimal number 0x04C11DB7. However, some CRC implementations use its reverse as 0xEDB88320.

Based on [9], this generator satisfies for 4096≤codeword≤12144 bits. The interval is equal to the size of Ethernet frame (512 – 1518 bytes). For this implementation, it has Hamming distance, HD = 4 meaning the generator are able to detect all 3 bits error and lower.

4.2 Generator Proposed by Castagnoli (CRC-32C)

The name Castagnoli refers to the author of [10]. Castagnoli, Braurer and Herrmann evolved technique of constructing dual code polynomial belong to Fujiwara. They built special purpose hardware to find out new generator to improve performance of IEEE 802.3. Several factorization classes of generator polynomial of size 24 and 32 were evaluated. The evaluation yielded four optimum classes of 32 bits polynomials. *First*, CRC-32/8 code whose factors into $(x + 1)^2$ and three distinct irreducible polynomials of degree 10. The generator equivalent to codes of data length 1023 bits with HD = 8. *Second*, CRC-32/6 is the code of CRC-32 whose factors into $(x + 1)^2$ and two distinct primitive polynomial of degree 15. This generator similar with CRC-32 code for data length 32767 bits and HD = 6. *Third*, generator CRC-32/5 whose consists of one polynomial of degree 32. It gives HD = 5 to 65535 bits data. And the last is generator CRC-32/4. It was resulted from 47000 such codes that factors into (x + 1) times a primitive polynomial of degree 31. This generator keeps HD = 4 but it covers at data words sizes in excess 64 Kb. This generator is represented as

$$g(x) = x^{32}+x^{28}+x^{27}+x^{26}+x^{25}+x^{23}+x^{22}+x^{20}+x^{19}+x^{18}+ x^{14}+x^{13}+x^{11}+x^{10}+x^{9}+x^{8}+x^{6}+1 \quad (5)$$

In the hexadecimal form is 0x1EDC6F41 or 0x82F63B78 in its reverse. RFC 3385 [11] proved this generator to be used in iSCSI (Internet Protocol Small Computer System Interface). Thus we choose the code as a candidate tc be used in CEH.

4.3 Generator Proposed by Koopman (CRC-32K)

Koopman did experiment to search a generator polynomial that covers larger data length [4]. He criticized the widely used generator polynomial that is IEEE 802.3. The standard only achieved Hamming Distance (HD) 4 for maximum packet length 12144 bits (Ethernet MTU). Whereas, theoretically it possible to detect HD = 6. The author searched a new generator polynomial that could to be used on HD = 6 for larger data length.

The evaluation was done for several generators including IEEE 802.3 and Castagnoli. The conclusion of the study is a 32 bits generator polynomial whose factors $(x+1)(x^3+x^2+1)(x^{28}+x^{22}+x^{20}+x^{19}+x^{16}+x^{14}+x^{12}+x^9+x^8+x^6+1)$. It constructs a full 32 bits generator represents as

$$g(x) = x^{32}+x^{30}+x^{29}+x^{28}+x^{26}+x^{20}+x^{19}+x^{17}+x^{16}+x^{15}+ x^{11}+x^{10}+x^7+x^6+x^4+x^2+x+1 \quad (6)$$

In binary form is 0x741B8CD7 or 0xEB31D82E of its reverse. The authors claimed the generator achieves HD = 6 for 16360 bits data length and HD = 4 to 114663 bits data length.

5. METHOD OF SELECTING GENERATOR POLYNOMIAL

We use two parameters to select the generator suitable for CEH based on previous work and our experiment. The two parameters are error detection capability and processing time to generate CEH in IPv6 packet transmission. The first parameter is analyzed by reviewing previous work on it and the latter analyzed by experiment. We configure small network to do the test bed in order to obtain particular data regarding processing time and delay transmission. The network is shown in figure 6.

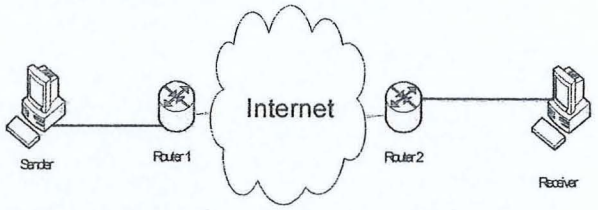


Figure 6 Topology of CRC Generator Polynomial Selection

We develop a program in Java that generates IPv6 packets with CRC extension header. The CRC code is generated from coverage area in figure 3. It is placed between IPv6 main header and TCP header and payload. The complete IPv6 packets are sent through the network. As common extension header, the CEH will not be processed in each router instead of in destination node indicated by destination address field. Another program is run in receiver side to compute the CRC code following figure 5.

6. RESULT AND DISCUSSION

In this section, we present result of our study on data length scope of the three candidates. Capability of generator polynomial to detect transmission error can be measured by its probability of undetected error [9], [10], [12]. This is summarized in table 1.

Based on table 1. HD = 4 and 6 whose available on the three candidates states obviously. This refers to the

existing widely generator polynomial which is Ethernet. It uses HD = 4 with minimum data length 4096 (512 bytes) and maximum 12144 (1518 bytes). The data length interval is used in standard IEEE 802.3 for all type of Ethernet. For CEH implementation, we chose to use HD = 4. It means all burst error with 3 bits and below is able to be detected by the generator polynomial.

Table 1 Summary of Data Length

Generator	Factor	HD	Data length (bit)
CRC-32E	32	3	$2^{32}-1$
		4	**4096 – 12144**
		5	512 – 2048
		6	204 – 300
CRC-32C	1,31	3	-
		4	**5276 – ($2^{31}-1$)**
		5	-
		6	210 – 5275
CRC-32K	1,3,28	3	-
		4	**16361 – 114663**
		5	-
		6	153 – 16360

Consider to RFC 2460 [12], the minimum MTU in IPv6 transmission over Ethernet is 1280 bytes or 10240 bits. This minimum length is able to be covered by CRC-32E and CRC-32C and it is not covered by CRC-32K. The maximum length of IPv6 MTU over Ethernet is 1518 bytes or 12144 bits. This value is also covered by CRC-32E and CRC-32C. However, for the CRC-32E this value is its maximum. Hence, it is difficult to use for the future because of increasing Ethernet MTU such as jumbo frame implementation. CRC-32C is suitable for larger MTU in the future.

Our experiment used topology in figure 6 to note processing time of each generator polynomial. Processing time of the first packet is different from the following packet. In case of the first packet, it runs full algorithm to generate a table 156-entry. Thus, time processing for the first packet is the biggest one. While other following packets consume fewer time to generate the CRC code. This is because it utilizes the preceding table lookup generated by the first packet.

We sent various sizes of IPv6 packet from sender to receiver: 64, 128, 25, 512, 1024, 1280 and 1492 bytes. The data documented are processing time in sender, receiver and total processing time. Processing time is the time required to generate CRC 32 code and insert it in IPv6 packet as extension header. With an assumption time of IPv6 packet generation is constant value, the processing times represent time to generate CRC code. The result is shown in figure 7, 8 and 9 respectively.

Figure 7 shows graph processing time (ms) vs packet size (bytes) at the sender side. The processing time increases with increasing packet size. This is inline with nature of CRC code that is linier code. Three generator polynomials show tight competition. All of packet size demonstrates small differences. However, the average processing time of CRC-32C is the lowest value that is 0.900 ms compare to CRC-32K 0.918 ms and CRC-32E 0.912 ms.

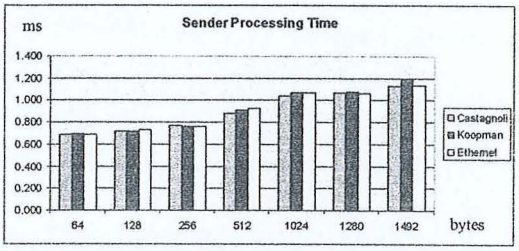


Figure 7 Sender Processing Time

Figure 8 shows processing time (ms) vs packet size (bytes) at the receiver side. The graph also demonstrates similar inclination with sender side. The processing time for the first IPv6 packet in receiver side increases with packet size increasing. However, receiver processing time is smaller than sender side. This is because in the receiver, there is no IPv6 packet generation.

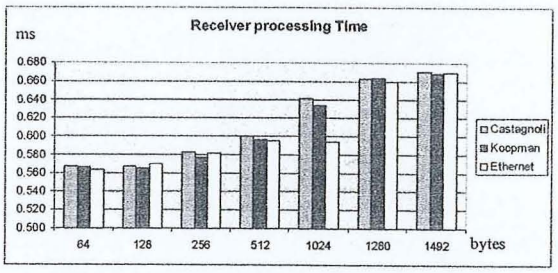


Figure 8 Processing Time in Receiver Side

The smallest average receiver processing time is belonging to CRC-32E that is 0.605 ms and 0.613 ms for CRC-32C and CRC-32K is 0.610 ms.

Total processing time of CRC code generation is shown in figure 9. Similar to previous processing time, the figure also illustrates that the three candidates are homogeny. It means the three generator polynomials are applicable in CEH from processing time point of view. Hence, capability of error detection that is analyzed in early of this section is a good way to determine which polynomial suitable for CEH.

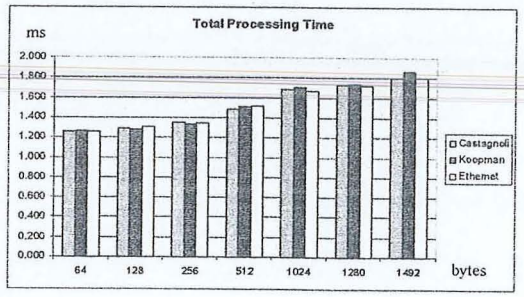


Figure 9 Total Processing Time

As stated earlier, the processing time is based on the first packet. The other following packet need smaller time processing to generate CRC code. This is because there is a CRC code stored on table from the first packet. Figure 10 demonstrates processing time for common Internet packet size that is 1518 bytes. The graph shows time processing for the following packet in exponentially decrease. The three generators shows their trend line enumerated below

$$CRC\text{-}32C = 0.7601x^{-0.9562}$$
$$CRC\text{-}32K = 0.7626x^{-0.9422}$$
$$CRC\text{-}32E = 0.7612x^{-0.9554}$$

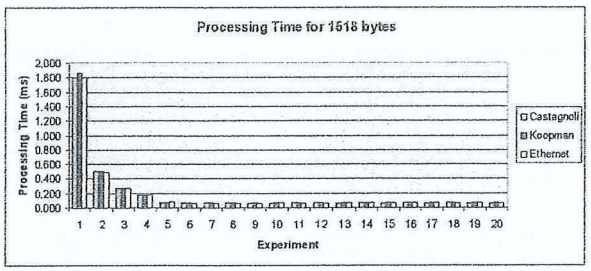


Figure 10 Time Processing for 1518 bytes packet size

The three exponential trend lines demonstrate that CRC-32C has smallest coefficient and power. This means the CRC generator polynomial has the best trend line. The processing time decreases toward smallest value.

7. CONCLUSION

We proposed CRC Extension Header (CEH) as a new mechanism to reduce duplicate error detection in intermediate node. CEH is applied to perform error control in the Network layer. With Network layer error control, we can eliminate error control in each intermediate node which possible to reduce the transmission time. CEH generation needs a generator polynomial. This paper described CEH and how selection of most suitable CRC-32 code generator polynomial for CEH was done. Among the generator used in existing Ethernet (CRC-32E) and two other generator suggested by Castagnoli (CRC-32C) and Koopman (CRC-32K), CRC-32C showed the best result. Its data length range covers larger data and at the same time the minimum data length also smaller than minimum MTU of IPv6 packet. The trend line of CRC-32C has smallest coefficient. We then conclude that CRC-32C suggested by Castagnoli is the most suitable generator polynomial for CRC extension header. Our future works will be implementing the CEH and analyzing its efficiency in handling the error detection in the high speed network.

REFERENCES

[1] F. Braun and M. Waldvogel, *Fast Incremental CRC Updates for IP over ATM Networks*, High IEEE Workshop on Performance Switching and Routing, 2001, pp. 48 – 52.

[2] Supriyanto, Raja Kumar Murugesan, Rahmat Budiarto, Sureswaran Ramadass, *Handling Transmission Error for IPv6 Packets over High Speed Networks*, Proceedings of the 4th International Conference on Distributed Framework for Multimedia Applications (DFMA 2008), Penang, 21-22 Oct., 2008, pp. 159-163.

[3] Castagnoli, G, Ganz, J. & Graber, P. *Optimum cycle redundancy-check codes with 16-bit redundancy*. IEEE Transactions on Communications, Vol. 38, 1990, pp. 111-114.

[4] Koopman, P. *32-bit cyclic redundancy codes for Internet applications*. Proceeding of International Conference on Dependable Systems and Networks, 2002.

[5] Martin Stigge, Henryk Plötz, Wolf Müller, Jens-Peter Redlich, *Reversing CRC Theory and Practice*. HU Berlin Public Report, 2006.

[6] Ross N. Williams. *A Painless Guide to CRC Error Detection Algorithms*. ftp://ftp.rocksoft.com/papers/crc_v3.txt, 1996.

[7] Sarwate, D. V. *Computation of Cyclic Redundancy Checks via Table Look-up. Commun. ACM*, Vol. 31, 1988. pp. 1008-1013.

[8] ANSI/IEEE Standard for Local Area Networks, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*. 1984.

[9] Fujiwara, T., Kasami, T. & Lin, S. *Error detecting capabilities of the shortened Hamming codes adopted for error detection in IEEE Standard 802.3*., IEEE Transactions on Communications, Vol. 37, 1989. pp. 986-989.

[10] Castagnoli, G., S. Brauer, and M. Herrmann, *Optimization of cyclic redundancy-check codes with 24 and 32 parity bits*. Communications, IEEE Transactions on, 1993. 41(6): p. 883-892.

[11] Shienwald, D. at. al. *Internet Protocol Small Computer System Interface (iSCSI) Cyclic Redundancy Checks (CRC/Checksum Considerations*. RFC 3385. The Internet Society.

[12] Deering, S. and R. Hinden (December 1998). *Internet Protocol Version 6 (IPv6) Specification*. RFC 2460. The Internet Society

CRC Extension Header (CEH): A New Model to Handle Transmission Error for IPv6 Packets over Fiber Optic Links

Supriyanto
Electrical Engineering
University of Sultan
Ageng Tirtayasa
(UNTIRTA)
Cilegon, Indonesia
supriuntirta@yahoo.com

Rahmat Budiarto
School of Computer
Sciences
Universiti Sains
Malaysia (USM)
Penang, Malaysia
rahmat@cs.usm.my

Raja Kumar Murugesan
National Advanced IPv6
Centre of Excellence
Universiti Sains
Malaysia (USM)
Penang, Malaysia
raja@nav6.org

Sureswaran Ramadass
National Advanced IPv6
Centre of Excellence
Universiti Sains
Malaysia (USM)
Penang, Malaysia
sures@nav6.org

Abstract—**The escalating growth of web based services has led to the rapid growth of the Internet. This ever increasing growth of the Internet has led to the depletion of Internet Protocol version 4 (IPv4) addresses. IPv6, the Next Generation Internet Protocol developed by IETF in the 1990s is getting wide spread in use replacing IPv4 to overcome its shortcomings including IP address exhaustion. IPv6 offers many advantages and features over IPv4 which could be utilized to handle some of the functions at layer 2 (Data link layer, with reference to the ISO Model) so that the performance of processing packets in term of operations can be improved. One such function of interest is the frame check-sum found in Ethernet headers at the Data link layer. In this paper we show how error handling can be done at the Network layer instead of at the Data link layer by utilizing the extension header feature of IPv6, and the characteristics of the fiber optic medium. Our emulation results show, handling error utilizing IPv6 extension header at the Network layer has smaller packet processing time than handling it in the data link layer that need to check error in every node. Smaller packet processing time means faster transmission of IPv6 packets.**

Keywords—**IPv6; FCS (Frame Check Sequence); CRC-32; Ethernet**

I. INTRODUCTION

To reduce the design complexity, Computer Networks follow a layered architecture [1]. Each layer is defined based on its preceding layer and has a set of well defined functions with clear cut boundaries. Also with layered architecture the implementation details of each layer is independent of other layers. When data are sent from one machine to the other, they pass through a series of layers before it reaches the other side. At each layer the packet is encapsulated with a header that contains control information to handle the data received at the other side by the corresponding layer. These headers are an overhead in terms of processing and if they can be handled efficiently it would save us processing time. Our focus and interest is on the error handling function done at the Data link layer. Error handling function would be costly in terms of processing as it needs be done literally at every node until a packet reaches the destination node. IPv6 offers a host of technical advantages and features such as large address space, fewer fields in the packet header, efficient hierarchical addressing and routing infrastructure, address auto configuration, built-in security, improved QoS and extensibility [2]. Our proposal is that the error handling function can be handled efficiently, reducing the overall packet processing time and thus improve the transmission of IPv6 packets. This can be achieved by utilizing the characteristics or capabilities of the communication medium used to transfer data, and by improving the existing error handling mechanisms at the lower layers.

In computer networks, errors normally occur at the communication medium called transmission errors caused by the medium due to interference such as White or Gaussian noise and Cross talk [3]. The most common approaches to detect the errors are parity check and cyclic redundancy check (CRC) techniques [4]. The errors that are detected by either parity or cyclic redundancy check can be corrected with two types of mechanism called Automatic Repeat reQuest (ARQ) and Forward Error Correction (FEC) [5].

At the lower layers the error check mechanism is applied in the Data Link layer. Ethernet uses the Frame Check Sequence (FCS) field to handle errors. This FCS mechanism is employed once at every source and destination hosts and twice at the intermediate nodes such as routers. In high speed networks this error check mechanism may waste time because of doing error checking many times at the intermediate nodes before the packet reaches the destination. This paper makes an effort to remove error detection from the Data Link layer and handle it at the Network layer utilizing the extensibility of IPv6 extension header. This paper also intends to encourage further discussion in this area.

The remainder of the paper is organized as follows. Section II gives an overview of Packet Transmission. In Section III, we present the methodology for handling error detection at the Network layer instead of at the Data link layer. Section IV discusses on how the proposed methodology would improve the transmission of IPv6 packets reducing the overall packet processing time. Finally, we present our conclusion in Section V.

978-1-4244-4683-4/09/$25.00 ©2009 IEEE

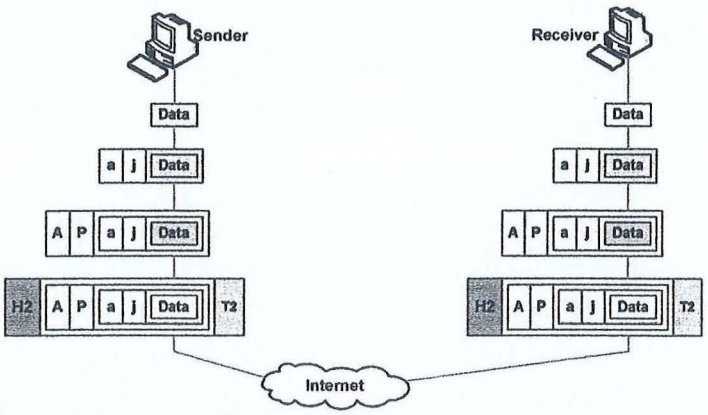


Figure 1. Encapsulation Process

II. PACKET TRANSMISSION

The ISO-OSI model and the TCP/IP model have been the two most popular network architectures that have been widely used. The ISO-OSI model has remained popular as a reference model for its simplicity and clarity of functions, while the TCP/IP was a more working model that is popularly used over the Internet. The reference made here applies to both the models. In terms of data communication, the data from the user on the sender side passes through a series of layers before it is transmitted over the communication medium to reach the other side. Data is encapsulated with control information added as headers at each layer. The process of encapsulation in data transmission is shown in Fig. 1. On the receiver side data received by the Physical layer goes up to the Application layer by discarding the header in each layer.

It can be seen from Fig.1 the original length of the data remains the same, but the length of the header increases with each layer. One of the quality parameters in data communication is the time needed for the data to reach the destination. Increased header length can cause increased delay time due to increased time needed to process the headers. If we can reduce these processing overheads the quality of packet transmission could be increased in terms of faster data transfer rates.

A. IPv6 Packet Format

An IPv6 packet consists of IPv6 base header, extension headers, and upper-layer protocol data unit. IPv6 base header is fixed in size and is of 40 bytes in length. The other functionalities needed are placed in payload as extension headers. Payload length may change due to the extension headers. Multiple extension headers can be used and use of extension header is optional. The IPv6 base header has 8 fields as shown in Fig. 2 [6]. Extension headers in IPv6 are a new way to deal with options that has substantially improved the routing process time. Extension headers are inserted into a packet only if the options are needed. They are processed in the order in which they are present. As the only extension header that is processed by every node on the path is the Hop-by-Hop Options header, it is placed first [6].

B. Error Check Mechanism

The popular error check mechanism that is currently being used at the link layer, especially in the Ethernet is the CRC method. The CRC is a type of hash function that is used to produce a small fixed-size checksum of a larger block of data such as a packet of network traffic. The checksum is used to detect errors after transmission. The CRC is computed and appended to a frame before transmission and verified afterwards by the recipient to confirm that no changes occurred to the data in transit.

The CRC is a popular method being used by most systems as they are simple to implement in binary hardware and easy to analyze mathematically, and are particularly good at detecting common errors caused by noise in transmission channels. In the Internet, currently, the error detection mechanism is placed in the Trailer part of the frame in the Ethernet as shown in Fig. 3 [7]. In our designed model we remove the checksum field or the CRC from the Ethernet frame and place it in the IPv6 extension header at the Network layer.

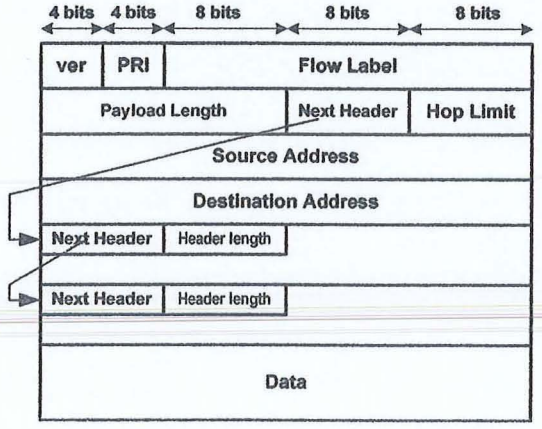


Figure 2. Header Format and Extension Header in IPv6

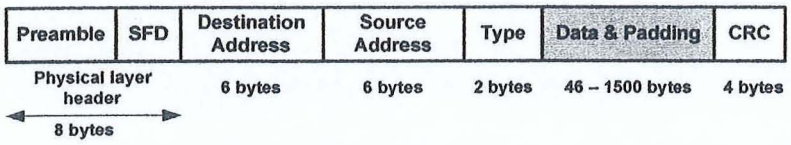


Figure 3. Ethernet Frame Format

III. METHODOLOGY

In this paper, we considered to place the error detection function in the Network Layer to check the whole IPv6 packet including header and payload. The error check mechanism used will be the same as CRC method that is currently being used with the existing systems. We defined CRC extension header (CEH) as a new IPv6 extension header to handle error detection for the entire IPv6 packet shown in Fig. 5. Validation of our proposed mechanism is conducted by a simulation of IPv6 packets transmission with CEH as error control mechanism in the network depicted in Fig. 4.

In order to get valid evaluation, we simulated not only CEH but also IPv6 packets transmission using FCS as error control in Data Link layer. We then compared performance of the two error control mechanisms including their processing time and delay.

In the first simulation, sender generates IPv6 packet and the corresponding CRC code to be inserted to the IPv6 packet as CEH. The packet with CEH is sent through a network with a topology as shown in Fig. 4 towards the receiver. The routers connecting the sender and receiver do not verify the CEH instead just determine the next path of the packet. The Receiver upon receiving the packet will verify CEH in its Network layer whether the packet is error free and then deliver to the upper layer. In case the packet received is erroneous, it will be discarded and wait for retransmission.

Second simulation represents the existing mechanism which is error control in Data Link layer. Sender generates IPv6 packet without any extension header inside. It is encapsulated in Data Link layer with header and trailer. The FCS in the trailer is actually CRC-32 code generated from the whole frame. First router of Fig. 4 will receive the packet and verify the CRC code inside. When the verification results no error it passes the packet to the Network layer to determine the next route. Bad packets will be discarded and the receiving node would wait for retransmission. This process will be carried out in all the intermediate routers connecting the sender and the receiver. From the two kinds of simulation of IPv6 packets transmission, we verified the following parameters that include processing time and delay. Results of the simulation are analyzed to verify the new error control performance.

As such, in the new model there is no error checking in the Data Link Layer. This means that error detection will be done only at the end hosts and will not be done in every intermediate node. We use fiber optic as the communication link to transmit data as it is faster and more reliable.

The newly designed mechanism of handling error detection in the Network layer using CEH with IPv6 packets was simulated and the results obtained are discussed in the next section. The results help us to study the performance of the designed mechanism in terms of packet processing time, error checking capability, and latency.

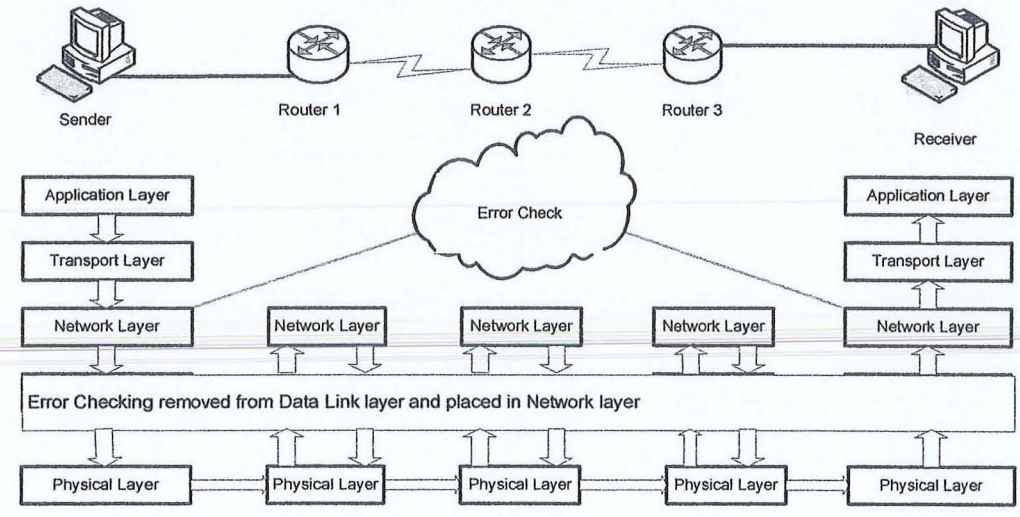


Figure 4. New Error Check Model

IV. DISCUSSION

Tere Parnell [8] states that Data Link layer encodes and frames data for transmission, in addition to providing error detection and flow control. Since the Data Link layer performs error checking, the same services need not be handled by the next higher layer. However, when a reliable medium is used, there is performance advantage by not handling error control in the Data Link layer, and instead handle it in another higher layer.

In terms of layering concept error control in general is handled at two different levels i.e. error control for upper layers and error control for lower layers. For the first, error control is handled by Transport layer. For IPv6 both TCP and UDP should apply checksum in its header. While for the lower layers error control is handled by the Data Link layer. The Data link layer uses CRC-32 to check for transmission errors caused by the transmission medium.

As both Data link layer and Network layer are lower layers where the protocol is between neighboring nodes we can place the error check mechanism in the Network layer instead of the Data link layer. With today's technology and faster routers this can be accomplished without compromising the data transfer rate. On the hosts it would also be efficient to do error checking at the packet level instead of at the frame level.

A. *Processing Time of CEH*

CEH utilizes the existing CRC-32 generator code that is standardized by IEEE 802.3 for Ethernet. Fig. 6 shows the relationship between processing time of IPv6 packets with CEH and size of IPv6 packets. It can be seen that the processing time for the IPv6 packet with CEH increases with packet length or size.

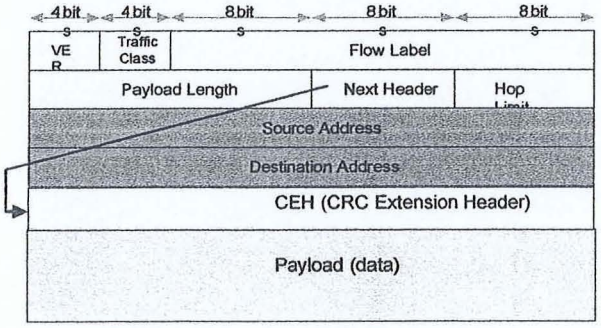


Figure 5. New IPv6 Packet with CRC32

The results also show that there are differences between processing time needed for the first IPv6 packet and the following packets. This scenario arises as CEH follows the fastest CRC-32 algorithm for computation in terms of table lookup [9]. It means that at the beginning, for the first packet processed a table will be created that will be used to generate the following CRC-32 code.

The biggest processing time is only for the first packet as seen in Fig. 7, and the correlation between processing time and packet sequence is negative exponential. This means processing time of successive packets is smaller both at the sender and the receiver. After the table with the pre computed CRC-32 codes generated, it is faster to generate CRC-32 code for the subsequent packets.

Fig. 8 shows the time delay in terms of IPv6 packet transmission for the existing IPv6 packets that uses FCS and our newly designed IPv6 packets that uses CEH. It can be seen that the IPv6 packets with FCS takes larger transmission time than packets with CEH as FCS needed to be processed in every intermediate node, especially twice.

Our emulation results also showed that there was no error in transmission of IPv6 packets with CEH over fiber optic links and no packet loss. To measure packet loss we set packet per second based on bandwidth capacity. Packet per second = bandwidth / packet size. So for the maximum size of a frame that is 1500 bytes based on Ethernet standards, we sent 8333 packet/second and observed the packet loss.

V. CONCLUSION

In this paper we have presented our new concept or method of handling errors at the Network layer instead of at the Data link layer, by utilizing the capabilities and features of the IPv6 protocol and the characteristics of high speed networks communication medium namely fiber.

The proposed method reduces the overhead in terms of header processing at the Data Link layer by removing the CRC field from its frame header and placing it in the extension headers of the IPv6 packet header in the Network layer. This proposed concept would enhance the performance of packet transmission in terms of faster data transfer rate, and thus enhance the performance of packet transmission.

ACKNOWLEDGMENT

This work was supported by the Research University Grant no.: 1001/PKOMP/817002 funded by Universiti Sains Malaysia.

REFERENCES

[1] A.S. Tanenbaum, *Computer Networks*, Fourth Edition, Prentice Hall of India. New Delhi, 2006.

[2] J. Davies, *Understanding IPv6*, Microsoft Press, Washington: Microsoft Press, 2003.

[3] N.F. Mir, *Computer and Communication Networks*, Pearson Education Inc., Crawfordsville, Indiana, 2006.

[4] S. Shukla and N.W. Bergmann, "Single bit error correction implementation in CRC-16 on FPGA." *In Proceedings of the IEEE International conference on Field-Programmable Technology*, pp. 319 -322, 2004.

[5] U. Demir and O. Aktas, "Raptor versus Reed Solomon forward error correction codes." *In the International Symposium on Computer Networks*, pp. 264 - 269, 2006.

[6] S. Deering and R. Hinden, "Internet Protocol Version 6 (IPv6) Specification," RFC 2460, IETF, December 1998.

[7] B.A. Forouzan, *Data Communications and Networking*, Fourth Edition, Tata McGraw-Hill Publishing Company Limited, New Delhi, India, 2006.

[8] T. Parnell, *Building High-Speed Networks*, First Edition, McGraw-Hill, 1999.

[9] T. Henriksson and L. Dake, "Implementation of fast CRC calculation." *In Proceedings of the Design Automation Conference, Asia and South Pacific*, ASP-DAC, pp. 563 - 564, 2003.

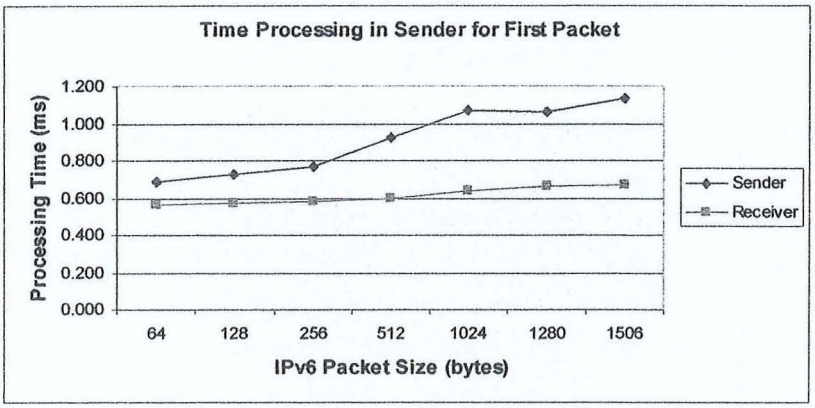


Figure 6. Time Processing for First Packet

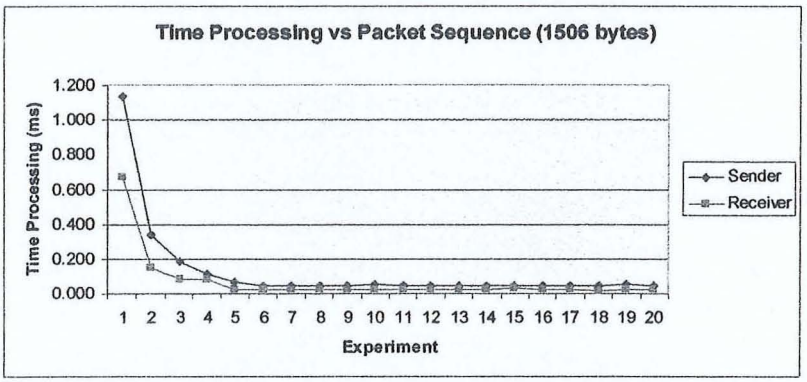


Figure 7. Time for CEH Processing

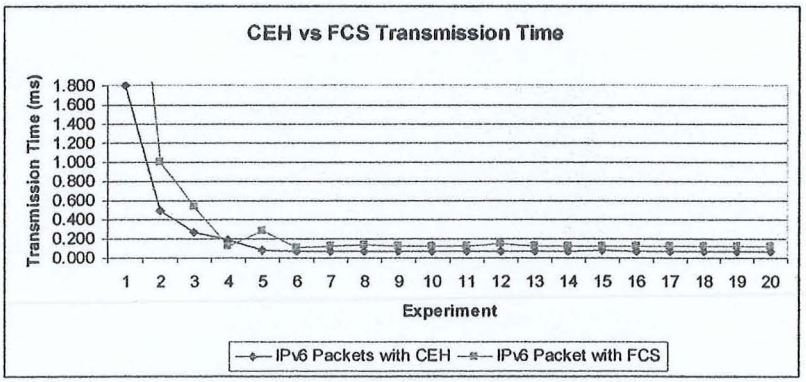


Figure 8. Delay of CEH Transmission

Exploiting IPv6 Extension Header to Handle Transmission Error for IPv6 Packets Over High Speed Networks

Supriyanto
Electrical Department, Sultan Ageng Tirtayasa University.
School of Computer Sciences
Universiti Sains Malaysia (USM)
supriuntirta@yahoo.com

Iznan H. Hasbullah
National Advanced IPv6 Centre of Excellence (NAv6)
Universiti Sains Malaysia
Penang, Malaysia
iznan@nav6.org

Rahmat Budiarto
School of Computer Sciences
Universiti Sains Malaysia (USM)
Penang, Malaysia
rahmat@cs.usm.my

***Abstract*—With the advances in IPv6 and its processing, verification and regeneration of cyclic redundancy check (CRC) in every node resulted in a bottleneck. This paper attempts to reduce the problem by decreasing redundancy of CRC calculation by taking advantage of one of IPv6 features, by introducing a new IPv6 extension header called CRC Extension Header (CEH) to do error checking in Network layer. The main component of CEH is CRC code generated from the entire IPv6 packet excluding one byte hop limit field. Our simulation result showed that eliminating verification and regeneration of CRC code in router successfully reduces transmission time of IPv6 packets transmission.**

I. Introduction

THE principle of IP network system is to transmit IP packets from one end point (source) to another point (destination). In order to transmit IPv6 packets from source to destination, the packets typically need to pass through router or routers. A router usually does a number of processing on the packet it received including error detection computation and packet forwarding decision. The router has to ensure each packet is free from error before forwarding it to the next router. A router might need to store the packets it received in its buffer in order to wait for the processing of prior packet to finish. Under certain condition, this may cause packets to be discarded and queuing saturation due to long time storing.

Manuscript received September 26, 2009. This work was supported in part by the USM-RU-PRGS under Grant 1001/PKOMP/832028 and in part by the Ministry of National Education of the Republic of Indonesia.

Supriyanto is with the Electrical Engineering Department, Sultan Ageng Tirtayasa University, Banten, Indonesia and School of Computer Sciences, Universiti Sains Malaysia. (e-mail: supriuntirta@ yahoo.com).

Iznan H. Hasbullah is with the National Advanced IPv6 Centre of Excellence (NAv6) Universiti Sains Malaysia, Penang, Malaysia. (e-mail: iznan@nav6.org).

Rahmat Budiarto is with the School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia (e-mail: rahmat@cs.usm.my).

Computer communications usually employ cyclic redundancy code (CRC) to do error detection in Data Link layer in the form of frame check sequence (FCS) field. In the protocol stacks, every intermediate node does the CRC verification and regeneration as shown in Figure.1. To make sure that the packet received by the router is free from error, the incoming port of Data Link layer of the router has to generate a CRC code based on the data received, and then comparing it with FCS field inside the packet. If the two CRC codes are the same, there is no transmission error and the packet will be delivered into the Network layer. Otherwise, the packet will be discarded and the receiver will wait for retransmission. Before the packet is sent to the next router, outgoing port of Data Link layer of the router has to regenerate a new CRC code and augments to the packet as new FCS.

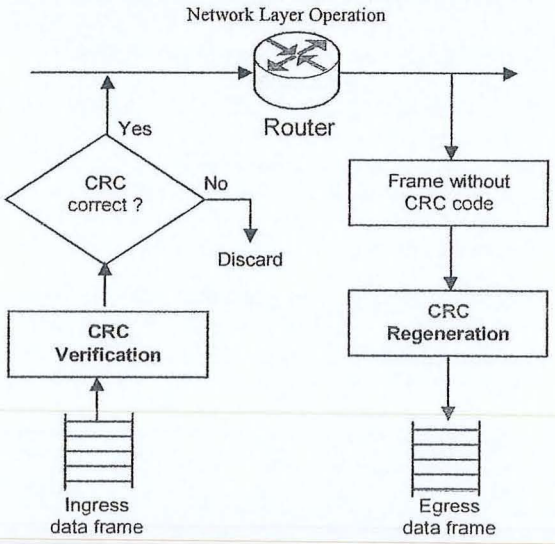


Figure 1 Duplication of CRC Calculation

With high speed network availability and fiber optic technology, verification and regeneration of CRC in each intermediate node is time consuming task. In addition, due to linearity of CRC code, bigger

packet size requires more time for computation. In fact, transmission error is almost zero error in very low bit error rate (BER) medium such as fiber optic [1]. Thus, error detection in Data Link layer (link by link error control) is likely to be an unnecessary redundancy. This paper intends to reduce the duplicate CRC calculation in IPv6 packet transmission system. It proposes eliminating CRC calculation in intermediate node and utilizing IPv6 extension header to detect transmission error in Network layer.

II. Related Work

Problems in IPv6 packets transmission over high speed network are due to overhead in the existing protocol stacks. The overhead is mostly because of non data processing including checksum and CRC calculation. Many researchers have tried to find out solution to the problem especially by reducing CRC calculation.

Only a few bits inside an IP packets changes during processing in each intermediate node. Most of the packet content remains the same especially the original data. In case of IPv4 packet, only 3 bytes which are one byte of TTL (time to live) field and two bytes of header checksum field will be changed. In IPv6 packet, only one byte hop limit field is changed in the forwarding node. Since there is only a small change of the IP header in the intermediate node, there is no need to check the whole packet for errors. The process is time consuming and unnecessary. The authors proposed a method called fast incremental CRC update. It distinguishes between CRC computation of changing field and non modified field. Sender generates both CRC code of two group's field and appends them to the packet.

In intermediate node, calculation is done only at the modified field and not in non modified field. Thus, CRC computation becomes faster because only few bits need to be checked. Overall CRC computation of a frame with IPv6 packet inside is 15 byes which are destination and source Ethernet address plus hop limit. This technique is effective to reduce CRC time processing. Intermediate node only requires computing 15 bytes instead of 512 bytes for minimal frame size. However, it still needs to do the calculation in each intermediate node. Frequency of CRC calculation is still similar with the existing system. It uses CRC-32E as generator polynomial to detect error in each node. CRC-32 is set to calculate data with size between 512 to 1528 bytes. Thus, utilization of CRC-32 to compute only few bits in this technique is not optimal [2].

A technique to reduce CRC calculation using fast CRC update was proposed in [3]. It implemented CRC parallel calculation. It separated CRC calculation of modified field and unmodified field. Calculation is only done on modified field, and then combines the CRC code with original CRC code of non modified field. This is done in intermediate node instead of final receiver. This technique also still calculates CRC in each node using CRC-32E. The difference with the previous method is the location to do combination between CRC code of modified field and the one of non modified field.

The latest research was reported in [4]. The authors suggested out of order incremental CRC computation. The method utilizes the fragmentation process in IP packet transmission. Traditional protocol splits a message into segments. Sender transmits each segment in different order and it may go through different network path. Usually, the receiver stores the segments until it received a complete message. Then, it processes the message including CRC calculation. Accordingly, it needs more memory to store the segments, requires long latency and extra bandwidth.

III. CRC Extension Header

Based on the identification of existing error control mechanism drawbacks in Section 1, we propose CRC (cyclic redundancy check) Extension Header (CEH) for error detection and correction in Network layer. CEH is a new extension header that is proposed to reduce bottleneck in IPv6 packet transmission over high speed networks by eliminating CRC verification and regeneration in each and every router as shown in Figure. 1. This idea is entirely different with the existing method that conducts error detection and correction in Data Link layer. This new mechanism does not require the Data Link layer to verify and regenerate CRC code for error detection. Verification of CRC code will be done at Network layer of the destination node by processing CEH inside the IPv6 packet received. Routers just need to process IPv6 packet at their Network layer as usual which is simply a forwarding decision.

There are three major rationales of this idea, firstly, optimally utilizing one of the IPv6 features especially extension header. IPv6 offers opportunity to discover new extension header for IPv6 packets transmission improvement in the future and it is not limited to 40 bytes only. In addition, applying a new extension header in the current network will not disturb the current IPv6 network operation. Secondly, very small possibility of transmission error occurring on fiber optic medium made it possible to bypass error detection function in higher layer. Thus, the duplication of CRC calculation could be eliminated from the network. Thirdly high transfer rate of current underlying technology especially gigabit Ethernet that is able to transmits more than 10 Gbps currently. Eliminating error detection process in intermediate node will decrease round trip delay from sender to receiver. Hence, low latency of IPv6 packets transmission over high speed network could be achieved.

The CEH is a combination of the advantages brought by IPv6 features and the advance of existing underlying technology. Since all of existing extension header already had a specific function, designing CEH cannot make use any of existing extension

header, but instead need to define a new extension header. It also still utilizes the advantages of the current error control and at the same time avoids its weaknesses. It uses the widely used error detection mechanism which is CRC-32 to detect transmission error. To obtain an optimal result, we select an appropriate generator polynomial to be implemented in CEH. In term of error correction, CEH applies retransmission procedure to overcome erroneous packet such as Automatic Repeat reQuest (ARQ). Details of CEH format and mechanism design will be presented in the next section.

A. Format of IPv6 with CRC Extension Header

There is no standard for IPv6 extension header format. The format of CEH follows the draft of generic IPv6 extension header (GIEH) proposed in [5] as shown in Figure. 2. However, it is not exactly similar to GIEH because CEH has an exact length of 32 bits as size of CRC-32. Thus, header extension length is fixed and two fields of the GIEH are omitted in CEH which are header extension length and specific type. Next header is an 8 bits indicator to point to other extension header following CEH. As a new extension header, CEH has not obtained specific number from Internet Assigned Numbers Authority (IANA). The rest of CEH format is allocated for future use instead of specific extension header type.

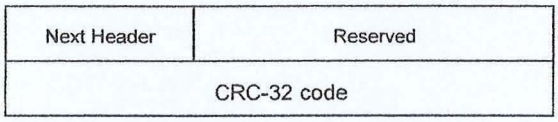


Figure 2 CRC Extension Header Format

B. CRC Extension Header Processing

CEH was designed to avoid CRC verification and regeneration in every node which resulted in more processing time. To do this, it follows typical extension header creation. It is generated by the sender and will be processed by the destination node. In an IPv6 packet, CEH is placed after destination address field before upper layer payload. If the packet has extension header chain, it is located in the last sequence after hop by hop option, routing header and fragmentation extension header. In the case where destination option extension header existed, it can be placed before or after CEH or both of them. This is because CEH will only be processed by destination node indicated by destination address. The IPv6 packet indicates CEH by next header field directly in the case where there is no extension header other than CEH. However, indicator value of CEH has not been assigned by IANA.

CEH is generated in sender side and then inserted into IPv6 packet between IPv6 header and payload. The main field of CEH, the CRC code, is generated from the entire IPv6 packet with the exception of the hop limit. The reason for excluding the hop limit field in the CRC generation is to avoid the changes of CRC code after the packet pass through each forwarding device. Hop limit is one byte field of IPv6 that will be reduced by one in every router. Reducing the value of hop limit in the IPv6 packets transmission is similar to time to live field of IPv4. Thus, it is understood that the changes is not an error.

C. CRC-32 Code Generation of CEH

CRC code generation requires a generator polynomial. Generator polynomial used in CEH will be selected from three candidates which are generator polynomial standardized in IEEE802.3 [6], generator proposed by Guy Castagnoli [7] and the one proposed by Philip Koopman [8]. Generation process of CEH follows the usual algorithm used in the Data Link layer which is already adapted for Network layer as shown in Figure 3.

When CEH is created in an IPv6 packet, the extension header field is first initialized to zero except the next header field. The sender generates CRC-32 code and then inserts the code into the CRC-32 code field of CEH. A complete IPv6 packet includes main header with next header value indicating the type of CEH, extension header which is CEH and payload. The packet is ready for transmission along the network path until it reaches its destination. The next section is regarding verification algorithm of CEH in destination node.

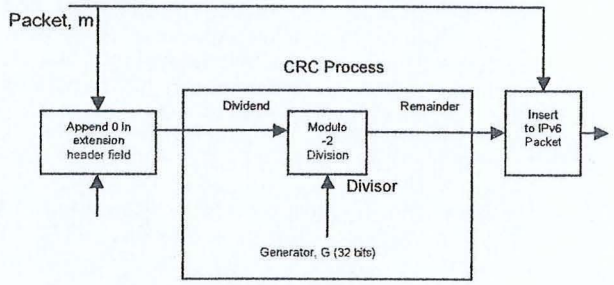


Figure 3 CEH Generation Process

D. CRC Extension Header Verification

Verification of CEH will be done at final destination following algorithm depicted in Figure 4. Once the receiver receives an IPv6 packet, it will check the next header field of the packet. When the value indicates there is CEH inside the packet, it extracts CEH and generates a new CRC code from the entire IPv6 packet but omitting the hop limit field.

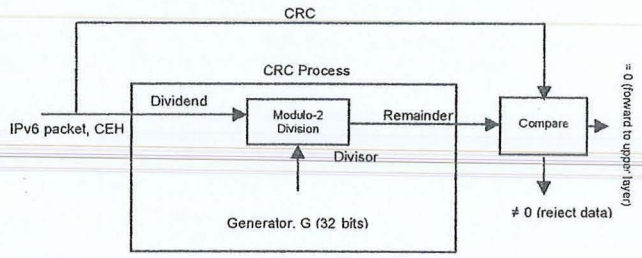


Figure 4 CEH Verification Process

The CRC code generated in receiver will be compared to CRC code inside IPv6 packet which is CRC code field of CRC extension header. If the two CRC code is the same, there is no transmission error in the IPv6 packet received. The receiver then forwards the packet into higher layer. Otherwise it will discard the packet and wait for retransmission.

IV. Simulation

CEH is a new concept of error detection for whole IPv6 that done in Network layer. To verify the acceptability of the new concept we did a simulation in a network topology depicted in Figure 5. The simulation has two scenarios: transmission of IPv6 packets with CEH as error control in Network layer and transmission of IPv6 packet with FCS as error control in Data Link layer. The first simulates the new concept and the later simulates the existing system. The two simulation scenarios yield an adequate data to justify performance of the new error control compared to the current error control.

The topology in Figure 5 has five nodes: sender, router 1, router 2, router 3 and receiver. All nodes are PC with Core 2 Duo processors that were configured as mini IPv6 network. Sender is an end system installed with IPv6 packets generator program in JAVA that is able to generate various type of IPv6 packets required in the simulations which are IPv6 packet with CEH and IPv6 packet without CEH. Router 1, 2 and 3 represent intermediate system of the network. In the first simulation, they just forward the IPv6 packets with CEH rather than checking each and every packet for error. The intermediate system in the second simulation performs error checking for each packet before forwarding the packets to next path. The last node is the receiver which is a PC installed with the same program to verify all packets received.

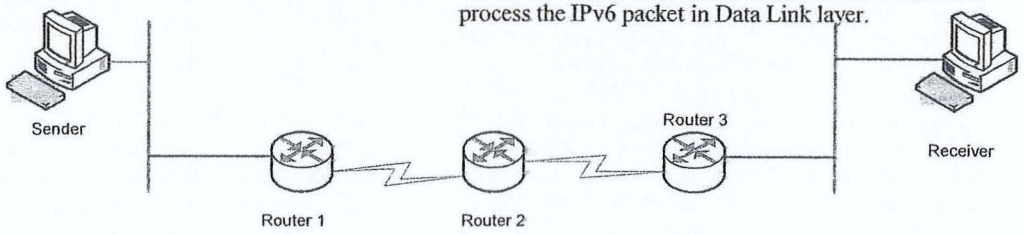


Figure 5 IPv6 Network Topology

The sender generates IPv6 packet by adding IPv6 main header to a TCP segment received from upper layer. The Network layer then generates CRC code from both IPv6 header and payload using algorithm in Figure 3. The CRC code obtained is inserted as CRC extension header into the IPv6 packet. Data Link layer in the first scenario just adds the link layer header. There is no trailer in the Data Link layer frame that usually does the error control task. The frame without frame check sequence (FCS) generated by the sender is then transmitted through the simulation network.

When the IPv6 packet reaches intermediate nodes which are router 1, 2 and 3, no CRC code calculation will be performed by the routers for error detection. The ingress frame is just verified for the frame header by incoming port of router and passed to Network layer to determine the next path (forwarding process). The egress frame also does not have Data Link layer trailer. It just obtains link layer header.

The only node that will verify the IPv6 packet is destination node (receiver) indicated by destination address field of the packet. The verification follows algorithm in Figure 4 to check whether the packet it received is free from error or not. Thus, receiver node's task are to receive the packet in Data Link layer, release the Data Link layer header without computing the CRC code and to deliver the packet to Network layer to do verification.

In the second simulation which is simulation of current error control mechanism, the sender generates IPv6 packet without extension header. The packet is then encapsulated by Data Link layer by adding Data Link layer header and trailer. Thus we called the frame as IPv6 packet with FCS. Each intermediate node will receive the packet and process it as usual. They process the Data Link layer header including CRC code computation to detect transmission error for corresponding hop. The only packet that is justified as free from error packet will be delivered to Network layer to do forwarding process without processing any extension header. Before the IPv6 packet is forwarded to the next hop, the packet will be encapsulated again in Data Link layer of the outgoing port of the router including CRC code regeneration.

Receiver node in the second simulation captures the IPv6 packet that addressed to it and the first packet processing is Data Link layer header and trailer verification including CRC code computation. Then, the correct packet will be passed to Network layer to do the next process which is IPv6 header verification. Otherwise, discard the erroneous packet and wait for retransmission. Receiver also notes the time needed to process the IPv6 packet in Data Link layer.

From the two simulations, we measure two main metrics: processing time and error detection capability. Processing time is packet processing time in a node such as CRC code computation and packet generation. This is very important to know the network latency which is the most important network performance. As error detection, it is also important to know the new error detection capability to detect transmission error.

V. Result and Discussion

As mentioned in Section 4, there are two metrics used in this paper; processing time and error detection capability. This section presents result of the two

metrics of the simulations and analyzes the result to obtain justification that the new error control mechanism is acceptable.

Processing time in sender is time required to generate IPv6 packet with CEH. While processing time in receiver is time required to verify IPv6 packet received. Experiments of IPv6 packet with CEH transmission using vary packet size from 64 bytes till 1500 bytes yielded result as shown in Figure 6. The Figure shows relation between total processing time and packets size both in sender and receiver. Both packet with CEH and packet with FCS require certain amount of time to process the packet. The processing time increases with the increase in packet size. This is because CRC code generation was done byte per byte of the packet. Thus more bits in a packet means longer time to do processing of IPv6 packets. Total processing time means the total processing time in sender and receiver.

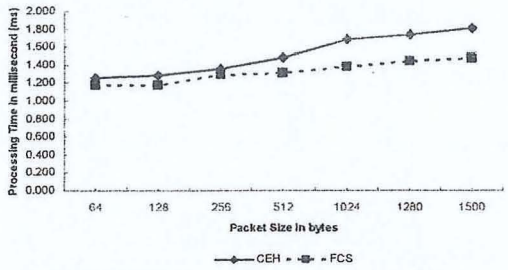


Figure 6 Processing Time vs. Packet Size

Figure 6 shows average total processing time of all packet size for transmission of IPv6 packet with CEH is 15 % higher than transmission of IPv6 with FCS. This is due to IPv6 packet with CEH generation in Network layer shown in Figure 3 is more complex than FCS generation in Data Link layer. In one hand, to generate CRC code from the whole IPv6 packet, it is required to exclude hop limit and then inserted the CRC code as extension header. In another hand, FCS generation just divided the whole frame with generator polynomial and appended the CRC code in the last part of frame.

However, transmission of IPv6 packet exploiting CEH as error detection in Network layer has eliminated CRC code calculation and regeneration in router. This causes faster processing of IPv6 packet in router. It just processes the packet in Network layer. Hence, although processing time at end system is higher, total network latency of IPv6 with CEH transmission is extremely lower than IPv6 with FCS. Figure 7 shows correlation between network latency in millisecond (ms) and packet size in bytes. In the Figure, network latency of IPv6 packet with CEH transmission is 68 % lower than IPv6 with FCS.

Error detection capability of error control mechanism is the ability to catch transmission error during transmission from sender till receiver. Recent high speed network technologies have made the occurrence of transmission errors very rare. Employing more error detection tools is inefficient and also redundant. However, we cannot eliminate the tools entirely because if error occurred the communication or data will be corrupted. Proposing to exploit CEH as error detection tool is the correct way. It can reduce the redundant error control process but at the same time it does not ignore the error.

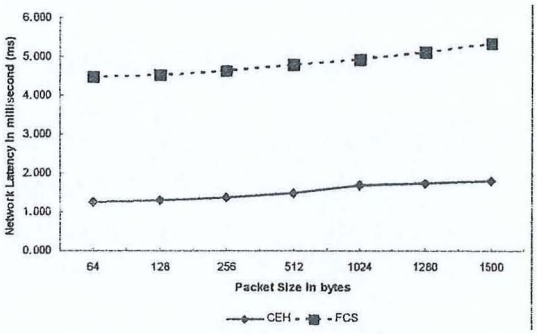


Figure 7 Network Latency

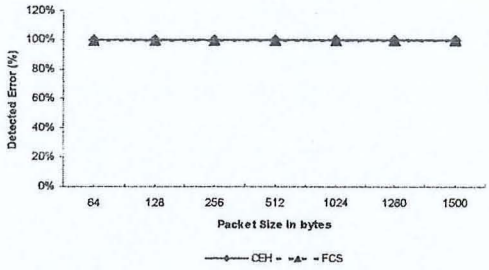


Figure 8 Error Detection Capability of CEH

Error is unpredictable. Thus an error control mechanism should be able to detect the error inside the packet transmitted and correct it. In order to know ability of CEH to detect transmission error, we send erroneous IPv6 packet with CEH. Figure 8 shows result of the experiment. The Figure illustrated correlation of amount of erroneous packet detected by receiver and packet size. The experiment showed that all erroneous IPv6 packets sent are successfully detected by the receiver.

VI. Conclusion

In this paper we have proposed a new concept of handling transmission error by exploiting IPv6 extension header. The new concept utilizes CRC extension header (CEH) as error control at Network layer and eliminates error control at Data Link layer of intermediate node. Typically, extension header is generated by the sender and is processed by the receiver. Thus, CEH that contains CRC code is also processed by the receiver.

Elimination of error control in Data Link layer means eliminating CRC code computation and regeneration in each and every router. This reduces total network latency on IPv6 packet transmission. The result shows the network latency decrease by 68 % compared with normal latency. It also reduces the Data Link layer frame size due to the elimination of 4 bytes of frame check sequence field. The proposed

error control mechanism also shows good ability to detect transmission error inside the transmitted packet. The experiment showed that all IPv6 packets sent with error are successfully detected by the receiver.

REFERENCES

[1] A.S. Tanenbaum, *Computer Networks*, Fourth Edition, Prentice Hall of India. New Delhi, 2006.

[2] F. Braun and M. Waldvogel, *Fast Incremental CRC Updates for IP over ATM Networks*, IEEE Workshop on High Performance Switching and Routing 2001. pp. 48 – 52.

[3] W. Lu and S. Wong, *A Fast CRC Update Implementation*, Proceedings of the 14th Annual Workshop on Circuits, Systems and Signal Processing, Veldhoven, The Netherlands, November 2003, pp. 113-120.

[4] J. Satran, *Out of Order Incremental CRC Computation*, IEEE Transactions on Computers, Vol. 54, No. 9, September 2005, pp. 1178 – 1181.

[5] S. Krishnan, et. al, *An uniform format for IPv6 extension headers*, Internet Draft IETF, 2008. http://ietfreport.isoc.org/idref/draft-krishnan-ipv6-exthdr/

[6] ANSI/IEEE Standard for Local Area Networks, *Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*. 1984. http://standards.ieee.org/getieee802/802.3.html.

[7] Castagnoli, G., Brauer, S. & Herrmann, M. *Optimization of cyclic redundancy-check codes with 24 and 32 parity bits*. IEEE Transactions on Communications, 1993, pp. 883-892.

[8] Koopman, P. *32-bit cyclic redundancy codes for Internet applications*, Proceedings. International Conference on Dependable Systems and Networks (DSN). 2002. http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn DBLP

Protecting Teredo Clients from Source Routing Exploits

Bassam Naji Abdullah Al-tamimi[1], Abidah Mat Taib[2], Rahmat Budiarto[3]

NAV6 Centre, Universiti Sains Malaysia
11800 USM, Pulau Pinang, Malaysia
[1]altamimi_net@yahoo.com
[2]abidah@nav6.org
[3]rahmat@nav6.org

***Abstract*— Tunneling techniques such as configured tunnel, 6to4, ISATAP and Teredo are common mechanisms in the early deployment of IPv6 to connect between two isolated IPv6 LANs or hosts by using the IPv4 infrastructure. We focused on Teredo tunnel as it allows users behind NATs to obtain IPv6 connectivity. Teredo tunnel has been designed to encapsulate IPv6 packet in UDP using IPv6-in-UDP-in-IPv4 technology. Though, Teredo tunnel raised some security threats including source routing exploits. This paper describes source routing exploits at the Teredo client and proposes a Teredo Client Protection Algorithm (TCPA) as an alternative mechanism to protect Teredo clients from IPv6 routing header risks. Since source routing in the IPv6 header could be exploited by either external or internal attackers, we believed our TCPA algorithm plays an impact in preventing potential attacks. TCPA is based on the filtration principle of matching. It operates on the Teredo client to deny the IPv6 packets which have routing header addresses unless the user allows these addresses traverse through it. The TCPA was implemented as a simulation in a real environment and the results showed that the proposed method is efficient and its logic sounds enough to protect Teredo client from attackers.**

***Keywords*— IPv6, Teredo tunnel, Teredo Client Protection Algorithm (TCPA), Routing header.**

I. INTRODUCTION

The IPv6 protocol was invented several years ago by the Internet Engineering Task Force (IETF) to be successor of IPv4. Many hosts and networks have been upgraded to support IPv6. However, there are still many existing IP networks supporting IPv4. Due to cost and application compatibility constraint, transition to IPv6 may occur in several phases which may begin with an isolated host or network migrating to IPv6 [1]. To ensure connectivity between these hosts and networks, tunneling these two sites over IPv4 is the most convenient way. This involves encapsulating an IPv6 packet inside an IPv4 packet in order to enable the encapsulated packet to be transmitted over the IPv4 infrastructure. This technology is called IPv6 over IPv4 tunneling [2].

Generally, transition mechanisms can be divided into dual stack, tunneling and translation [3]. However, in the early deployment of IPv6, tunneling techniques and dual stack hosts are common mechanisms use in the network configuration. Among the tunneling techniques available are manually configured tunnel and automatic tunnel such as 6to4, ISATAP, Teredo and Tunnel Broker. IPv6 allows end-to-end connection but at present, many Internet users can access Internet only through NATs (Network Address Translation). Although IPv6 does not support NAT, during the transition period, this scenario needs to be considered. Therefore, Teredo tunnel has been designed to allow users sitting behind NATs to access the IPv6 network by tunneling IPv6 packets in UDP.

Teredo tunneling is an automatic tunneling mechanism which was originally developed by Microsoft. It is a service that enables hosts located behind one or more IPv4 NATs to obtain IPv6 connectivity by tunneling packets over UDP [4]. So, the IPv4 network is treated as the link layer, and the existing IPv4 routing mechanism is utilized to forward IPv4-in-UDP-in IPv4 encapsulated packets.

However, the Teredo tunnel raises some security concerns, which might cause security gaps. Recently, IETF published a draft [5] including Teredo bypasses network security, challenges in inspecting and filtering content of Teredo data packets, and the increased exposure due to Teredo tunneling. Among the problems include source routing after Teredo client, which has motivated us to study the issue and workout for the possible solution.

In this regard, if IPv6 host specified source routing in IPv6 packet to forward behind the receiver Teredo client and the client in turn forwards it to the specified next hop. This process may be unanticipated and against networks administrators policies and may have bypassed network-based source routing controls [5].

This paper focuses on Teredo and its security threat namely source routing exploit in which discussion on the scenario and its proposed solution will be raised up.

In this paper, we begin with overview of the Teredo in section 2, followed with explanation on

978-1-4244-2312-5/08/$25.00 ©2008 IEEE

source routing threat in next section. Then, a proposed solution to the problem by using a Teredo Client Protection Algorithm (TCPA) will be discussed in terms of experimental setup and simulation result. Finally, we conclude the paper with conclusion and future work.

II. TEREDO TUNNEL OVER VIEW

The tunneling of IPv6 over IPv4 transition mechanisms such as the ISATAP and 6to4, does not typically work through the NATs [1]. Therefore, Teredo was introduced to enable the hosts located behind one or multiple IPv4 NATs to obtain IPv6 nodes connectivity by tunneling packets over UDP. The Teredo protocol was initially called the shipworm, based on a species of mollusk that digs holes in ship hulls, analogous to what the protocol does with NAT devices.

A. Teredo Components

The basic components of the Teredo architecture are Teredo client, Teredo server, and Teredo relay:

1) Teredo Clients: An IPv6/IPv4 node which supports a Teredo tunneling is called a Teredo client. A Teredo client establishes contact with a Teredo server to obtain the IPv6 address, so as to enable to reach other Teredo clients on the IPv6 Internet [4]. The Teredo clients send and receive Teredo IPv6 traffic tunneled via the UDP over IPv4 [4].

2) *Teredo Relay:* A Teredo relay server as the IPv6 router sited between IPv6 native networks (IPv6 Internet) and IPv4 networks (typically IPv4 Internet). The Teredo relay can also provide interoperability with hosts using other transition mechanisms such as 6to4 [4]. A Teredo Relay encapsulates IPv6 packets to Teredo clients in IPv4 UDP and decapsulates the IPv4 UDP packets sent from the Teredo client to the IPv6 network.

3) *Teredo Server:* A Teredo server is stateless and only has to handle a small part of the traffic between Teredo clients. It also assists Teredo clients to set up tunnels to IPv6 nodes [4].

The IPv6 traffic is encapsulated inside the IPv4 packet which has a protocol field in the header set to 41. Most NAT routers do not support (translate) protocol 41 because protocol 41 is not common as TCP, UDP and ICMP protocols. In this case, IPv4 encapsulated IPv6 traffic will not flow through these NATs. In order to overcome this problem, Teredo encapsulates the IPv6 packets as an IPv4 UDP message to allow the IPv6 traffic to flow through the various types of NAT routers [6, 7].

B. Teredo Addresses Format

A Teredo address consists of a Teredo prefix, a Teredo server IPv4 address, a Flag, an Obscured external port, and an Obscured external address. Teredo Addresses contain sufficient information for the relay to reach a client (see Fig. 1).

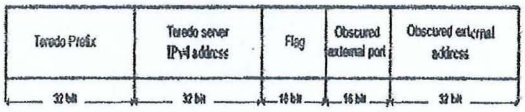


Fig. 1 Teredo address format

C. How teredo tunnel works

Fig. 2 illustrates an example to provide a clear view of how Teredo works.

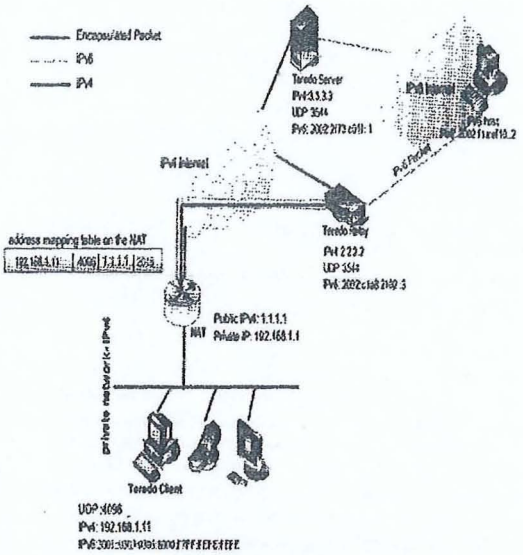


Fig. 2 Teredo tunneling architecture [8]

The first step in establishing communication between a Teredo client and IPv6 host is when the Teredo client sends a packet through the NAT to the Teredo server to obtain the Teredo address.

Assignment of the NAT occurs when the packet from a Teredo client is received; and the NAT automatically generates a new UDP port for the packet, and keeps the packet information in the mapping NAT table before sending it. This information includes the Teredo clients IPv4 address with its UDP, the NAT public IPv4 address and the new UDP allocated by the NAT.

Upon receipt of the packet which comes from the Teredo client, the Teredo server will conduct some calculations to assign a Teredo IPv6 address for the Teredo client. In the case where a Teredo client receives an IPv6 packet from an IPv6 host, the detailed steps are showed as follows:

In the packet transmutation phase, the IPv6 packet transmits from an IPv6 host to the Teredo client. This operation is executed through two phases. In the first phase, an IPv6 host forwards the packet to the Teredo relay while in the second

phase; the packet is handled by the Teredo relay before it is sent. This handling technique is called the encapsulation technique where the Teredo relay encapsulates the IPv6 packet within IPv4 and UDP.

The packet encapsulated format in the Teredo relay, consists of the following information:

- *Source IPv4 address* is the Teredo relay IPv4 address.
- *Source UDP port* is the Teredo relay UDP.
- *Destination IPv4 address* is the NAT public IPv4 address.
- *Destination UDP port* is the new UDP port (for the NAT).

The Teredo relay forwards the encapsulated packet to the destination IPv4 address (NAT) through the IPv4 Internet and then the NAT forwards the packet to the destination node according to the resident data in the NAT table.

III. Source Routing - Threats to Teredo Clients

All hosts which are located behind the NAT have IPv4 addresses and these hosts receive IPv4 traffic to deal with. Since IPv6 does not support NAT, the networks security systems that use the NAT deal only with the IPv4 security issues Fig. 3. One of the IPv4 security problems lies in source routing which the attackers can exploit to launch an attack against the network.

In IPv4, the source routing option has two forms of extension headers; either a strict source routing or a loose source routing header. The strict source routing can specify the exacting route to the packet in order to pass through to the final destination, and also list in its extension header up to 9 IP addresses. In contrast, the loose source routing option is capable to specifying one or more intermediate nodes addresses that a packet needs to pass through in order to reach a destination. In other words, rather than specifying a full path from the source node to the final destination, the loose source routing specifies just a sequence of landmark addresses to reach the destination [9].

The importance of the source routing lies in its use as a good diagnostic tool to verify network connectivity. However, it also constitutes a threat which attackers may exploit in order to bypass network security systems. Some security mechanisms such as Cisco IOS are proposed to handle such a problem. Easy solution involves discarding the packet that carries IPv4 source routing (strict/loose).

The source routing problem similarly occurs in IPv6 routing header where resembles the loose source routing problem in IPv4. Although some security mechanisms exist to deal with IPv6 packet with routing header, these mechanisms are unable to deal with IPv4 NAT security problems (see Fig. 4)

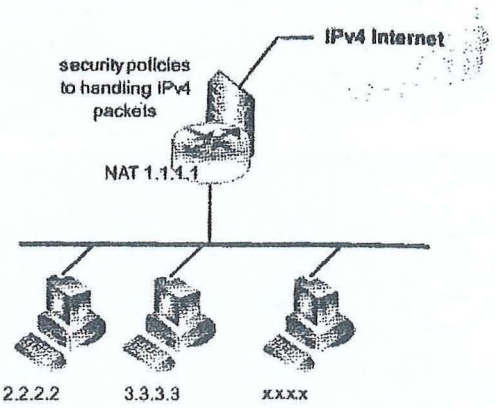


Fig. 3 IPv4 Network native

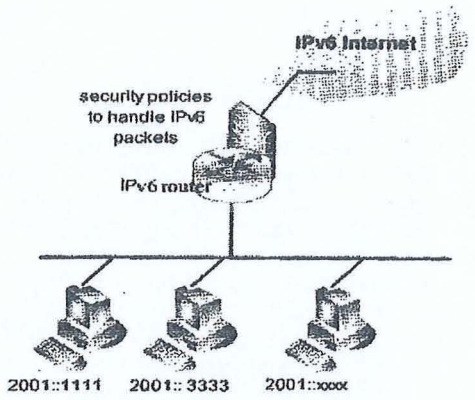


Fig. 4 Native IPv6 Network

In spite of allowing the dual stack hosts located behind IPv4 NAT to deal with IPv6, the Teredo tunnel raises the source routing problem where the attackers can exploit this security hole to launch their attacks against the Teredo client. In this scenario, attacks are launched via the IPv4 NAT using the IPv6 routing header, where the IPv6 packet reaches the Teredo client without being filtered by IPv4 security policies that reside on the boundary of IPv4 networks.

A. *How attackers exploits the source routing hole through the Teredo Tunnel:*

Attackers usually exploit the IPv6 functionality routing header type 0 to launch their attacks through the Teredo tunnel. Fig. 5 illustrates how these attackers can infiltrate the Teredo infrastructure.

Assuming that due to specified security reasons, these attackers are unable to reach directly to the Teredo-Client2. Still, they can find a way to bypass the security systems of the client by exploiting the routing header feature in IPv6. They may begin with crafting a packet which specifies the route to reach

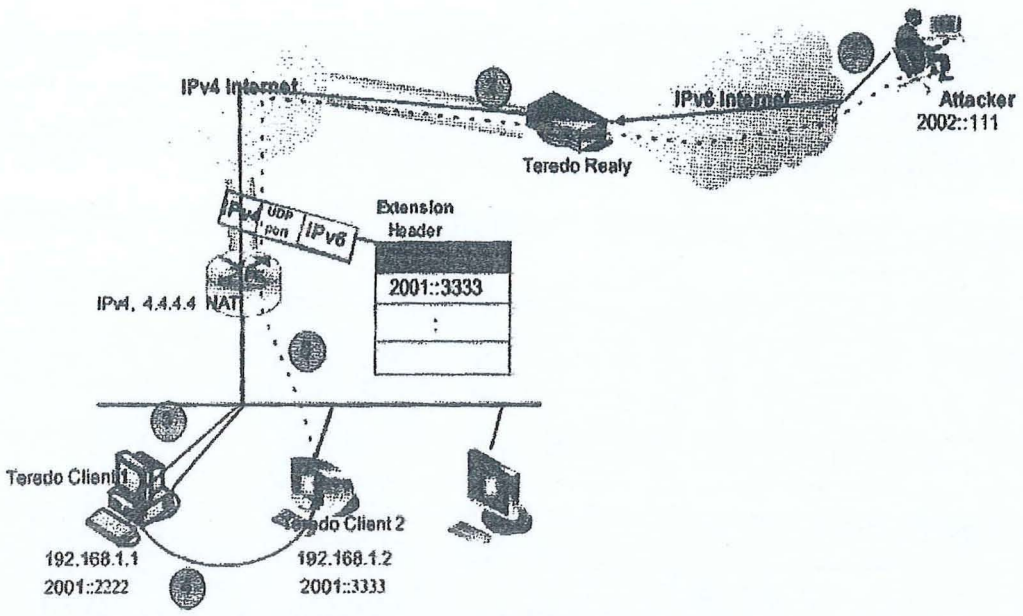


Fig. 5: The attackers' behavior on the Teredo infrastructure

the final destination. As shown in the illustration, they set up the packet to bypass the security filter by traversing through the T-Client1 first before being forwarded to T-Client2. If proper filtering is not being done at the end node, the attackers' packet will reach the supposed destination and potentially will do harm or trigger an attack.

To deal with this source routing exploitation, we have come up with a proposed algorithm to be placed at the end node (for instance T-Client2 in the illustration) for filtering packet with source routing specified in the routing header. Detail of this algorithm will be discussed in the next section.

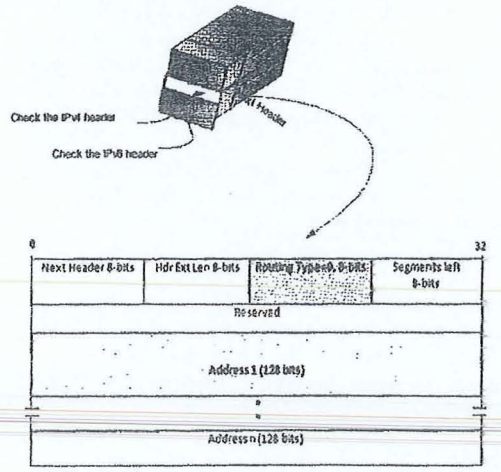


Fig. 6 IPv6 routing extension header

B. *IPv6 Routing Extension Header*

One method to solve this problem is by filtering the IPv6 header. Particularly, by check the Next header field in the IPv6 header to determine whether the routing header exists or not as illustrated in Fig. 6.

IV. Teredo Client Protection Algorithm (TCPA)

The proposed algorithm, shown in Fig. 7, aims to inspect the IPv6 packet; to determine whether it refers to the routing header or not. This process is very crucial to achieve our goal of protecting the Teredo clients from detours instigated by as these attackers.

- Src IP= IPv6 Source address for the sender.
- N-H= IPv6 Next Header field.
- R-H Type= Routing Header Type.
- Dst-RH= Destination address for Routing Header (RH).
- S-L= Segments Left for the source routing header (SRH). It refers to the number of the nodes which remains to be visited before arrival to the final destination.
- Value 43 in the Next header field indicates that the extension header is a routing header.

This algorithm is based on IPv6 filtering rules. When an IPv6 packet arrives at the Teredo client that holds the TCPA (Teredo Client Protection Algorithm), this packet will be subjected to filtration, according to IPv6 filtration rules. If the packet does not match the rules, the packet will be dropped. In case of matching, the next header of the

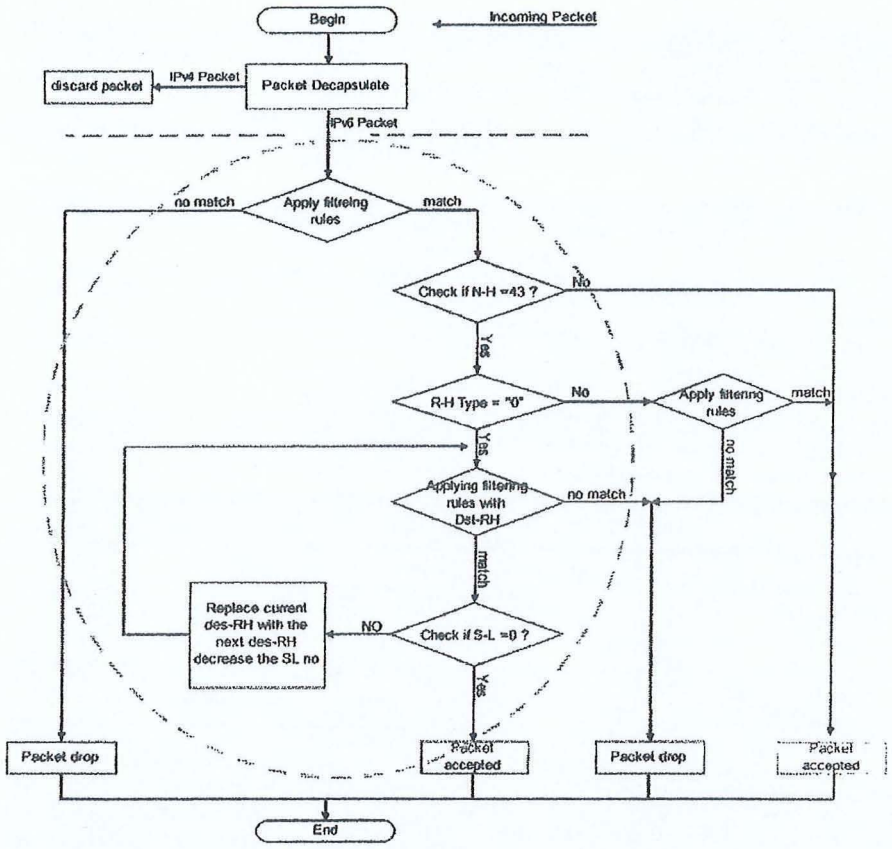


Fig. 7: Teredo Client Protection Algorithm (TCPA)

packet will be examined. If the next header field does not refer to the Routing header, the packet is accepted. On the other hand, if the next header field refers to the Routing header, the packet will be tested to determine the type of routing header. If the routing header is of type 2, the packet will be subjected to filtering, and if it matches with filtering rules the packet will be accepted, if it does not match, it will be dropped. If the type of routing header is 0, the packet is subjected to filtering with address of destination routing header (dst-rh). In this case, if the packet does not match the filtering rules, the packet will be dropped. But if it matches the filtering rules, a test is conducted to identify the segment left value. If this value equals zero, then the packet will be accepted, if it does not equal zero, the current destination routing header will be replaced with the next destination routing header, while at the same time the segment left value decreases. This filtration process is repeated continuously with the new destination header till the value of segment left field reaches zero.

A. *Packet Filtering Stages*

The filtration area in this algorithm (TCPA) is divided into five stages as illustrated in Fig. 8.

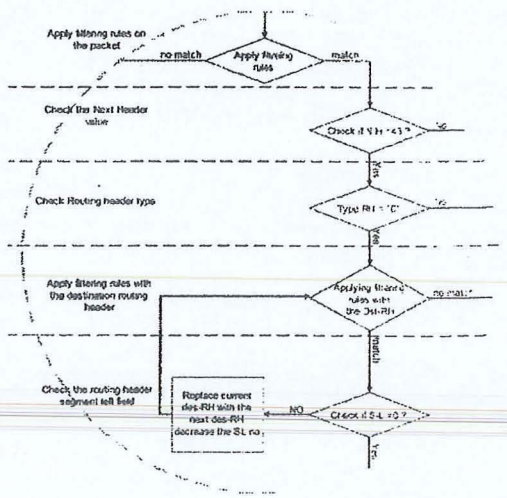


Fig. 8 Packet filtering stages

The use of algorithm can protect a Teredo client from risks associated with the routing header. As all the incoming packets of Teredo clients that have the routing header addresses will be subjected to TCPA checks, attackers cannot detour through the source routing hole.

V. The Experimental Set up Topology

The proposed method is simulated by using a Native IPv6 Network A and Native IPv4 Network B as a test bed. Fig. 9 illustrates the topology of the experimental test. The Network A has an IPv6 tunnel to the Network B. Basically, devices extant in the native IPv6 Network have an IPv6 addresses. In addition to this, there is a tunnel gateway on the boundary of the Network A to deal with IPv6 traffic. The Network B devices are supported by an IPv6 addresses that enable them to communicate with IPv6 networks (A) through the IPv4 Internet. Apart from this, Teredo clients in the Network B use the Teredo Client Protection Algorithm (TCPA) to check incoming IPv6 packets that consists the source routing header.

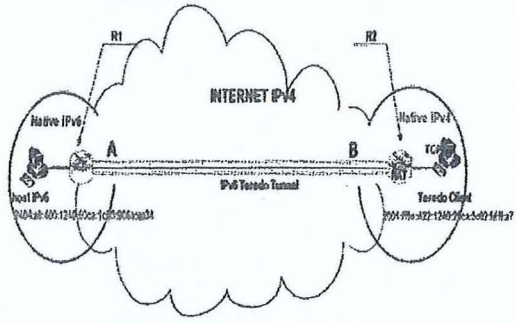


Fig. 9 The experiment test bed

In the Network B, the Teredo client operates on a Microsoft Windows XP operating system platform powered by Intel Pentium 4 duo core 2.00 GHz processor, memory 1024 MB, and Teredo address 2001:fffe:422:1240:20ca:3c02:fdff:a7. The Teredo client receives IPv6 packets from the IPv6 host and processes these packets according to TCPA.

VI. Simulation Results

Table 1 contains data referring to the time spent in the matching process. This process was conducted between routing header addresses that consist of the routing header type 0 and a list of addresses in the Teredo client. Therefore, the results of calculation time changed from time to time on the basis of several factors such as packets number and the number of routing header addresses in each packet. The results of the simulation shown in Fig.10, indicates an increase in time when the number of packets and number of routing header addresses increased.

In the Teredo client, the position of IP addresses which reside in the matching list is a very important factor in determining the time spent in packet processing. For instance, in type 0, if the TCPA found the first address did not match that on the list, processing operation will stop immediately, even if the packet contained a large number of routing header addresses.

Table 2 illustrates the average time spent on checking recipient packets while the average results in the Fig. 11 demonstrate a small increase in the delay time. In other words, when the number of next hops addresses increased the line of the time increased concomitantly.

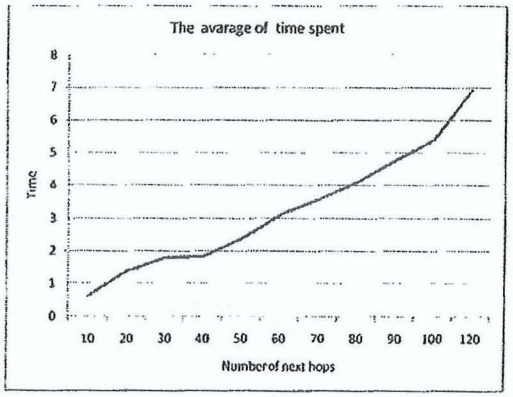


Fig. 11 The average of time spent

The results indicate that the TCPA algorithm is efficient in preventing source routing exploits as it uses efficient matching principle to either allow or drop the packets. In spite of the delay in the process of filtering, it is still useful and effective in preventing the detours caused by attackers.

VII. Conclusions and Future Work

In this paper we have described the Teredo tunnel mechanism and one of its potential security threat namely source routing exploits at Teredo clients. This paper has highlighted the threat scenario and proposed a Teredo Client Protection Algorithm (TCPA) as a prevention method. TCPA is based on the filtration principle of matching. It operates on the Teredo client by filtering incoming packets to deny the IPv6 packets that have routing header addresses unless the user allows these addresses traverse through it. TCPA was implemented as a simulation in a real environment and the results indicated that the proposed algorithm is efficient and its logic sounds enough to protect Teredo client from attackers. Unfortunately, network administrators are unable to perform IPv6 ping or network mapping if the Teredo client used TCPA unless all the addresses specified in the ping packet were authorized to traverse through it.

The IPv6 routing header type 2 that used in mobile IPv6 is not our main concern in this work. Thus, it needs to be studied in the future.

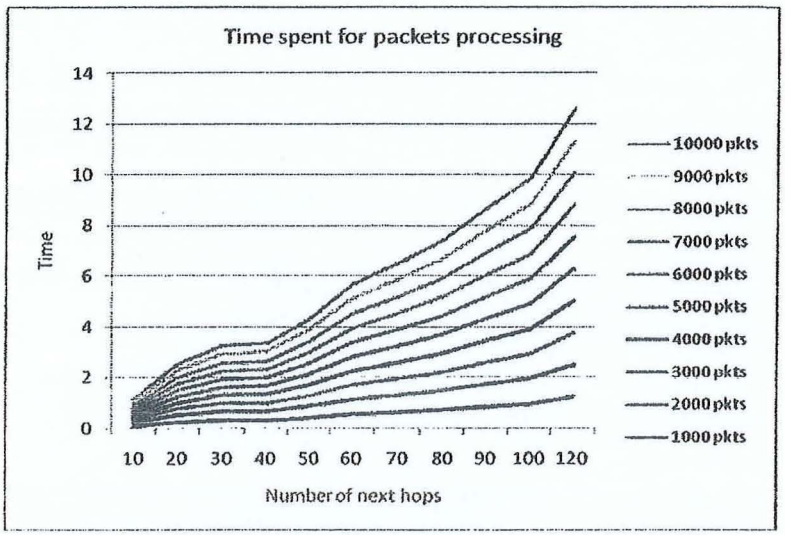


Fig. 10 The time spent on RH filtration

TABLE I
THE TIME SPENT ON PACKETS PROCESSING

NUMBER OF PACKETS	NUMBER OF ROUTING HEADER ADDRESSES											
		10	20	30	40	50	60	70	80	90	100	120
	1000	0.113	0.252	0.325	0.335	0.431	0.567	0.651	0.743	0.869	0.982	1.257
	2000	0.226	0.504	0.65	0.67	0.862	1.134	1.302	1.486	1.738	1.964	2.514
	3000	0.339	0.756	0.975	1.005	1.293	1.701	1.953	2.229	2.607	2.946	3.771
	4000	0.452	1.008	1.3	1.34	1.724	2.268	2.604	2.972	3.476	3.928	5.028
	5000	0.565	1.26	1.625	1.675	2.155	2.835	3.255	3.715	4.345	4.91	6.285
	6000	0.678	1.512	1.95	2.01	2.586	3.402	3.906	4.458	5.214	5.892	7.542
	7000	0.791	1.764	2.275	2.345	3.017	3.969	4.557	5.201	6.083	6.874	8.799
	8000	0.904	2.016	2.6	2.68	3.448	4.536	5.208	5.944	6.952	7.856	10.056
	9000	1.017	2.268	2.925	3.015	3.879	5.103	5.859	6.687	7.821	8.838	11.313
	10000	1.13	2.52	3.25	3.35	4.31	5.67	6.51	7.43	8.69	9.82	12.57

TABLE II
AVERAGE TIME SPENT

	NUMBER OF ROUTING HEADER ADDRESSES										
No. Addresses	10	20	30	40	50	60	70	80	90	100	120
Average. Δtx	0.6215	1.386	1.7875	1.8425	2.3705	3.1185	3.5805	4.0865	4.7795	5.401	6.9135

ACKNOWLEDGMENT

We would like to thank the NAV6 Centre of Universiti Sains Malaysia for the technical supports.

REFERENCES

[1] D. J. Hoagland, "The Teredo Protocol: Tunneling Past Network Security and Other Security Implications", Symantec. Principal Security Researcher Symantec Advanced Threat Research. 2007.

[2] R. Gilligan, and E. Nordmark," Transition Mechanisms for IPv6 Hosts and Routers", *IETF Request for Comments, proposed standard,* RFC2893,2000. http://www.ietf.org/rfc/rfc2893.txt

[3] E. CARMÈS, "The Transition to IPv6", The Internet Society (ISOC), 1775 Wiehle Ave.,Suite 102, Reston, Virginia 20190 USA 2002.

[4] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translation (NATs)", RFC 4380, February 2006.

[5] J. Hoagland and S. Krishnan, "Teredo Security Concerns", IETF, draft-ietf-v6ops-teredo-security-concerns-01, 2007.

[6] Microsoft, "Teredo Overview", 2007.

[7] J. Palet, C. Olvera, and D. Fernandez, "Forwarding Protocol 41 in NAT Boxes",draft-palet-v6ops-proto41-nat-03.txt October 2003.

[8] S.-M. Huang, Q. Wu, and Y.-B. Lin, "Tunneling IPv6 through NAT with Teredo Mechanism", presented at International Conference on Advanced Information Networking and Applications, 2005.

[9] G. Huston, "The ISP Column", Internet Society May 2007.

European Journal of Scientific Research
ISSN 1450-216X Vol.43 No.2 (2010), pp.192-203

http://www.eurojournals.com/ejsr.htm

Error Detection using CRC Extension Header for IPv6 Packets over High Speed Networks

Supriyanto
Faculty of Engineering UNTIRTA Indonesia and Univeriti Sains Malaysia
E-mail: supriyanto@ft-untirta.ac.id

Manjur Al Kolhar
Universiti Sains Malaysia
E-mail: m_kolhar@yahoo.com

Rahmat Budiarto
Universiti Sains Malaysia
E-mail: rahmat@cs.usm.my

Zainal A. Hasibuan
University of Indonesia
E-mail: zhasibua@cs.ui.ac.id

Abstract

IPv6 has come with a package of advantages including simple header format, very large address space and extensibility. However, IPv6 packets transmission still uses the traditional infrastructure of protocol stacks such as TCP/IP. Thus, the big advantages cannot be taken optimally. One of the limitations of TCP/IP is duplication of error detection code verification and regeneration in Data Link layer. Every router has to verify CRC code at incoming port and regenerate the CRC code at outgoing port before forward an IPv6 packet to the next router. With advance networking technology this is a time consuming task. This paper proposes CRC Extension Header (CEH) to do error detection in Network layer and replaces the current error detection in Data Link layer. In CEH, verification of CRC code is only done in the final destination indicated by destination address field of IPv6 header. Experimentation results showed network latency of IPv6 packets transmission decreases 68%.

Keywords: CRC, error detection, network latency, extension header

1. Introduction

Nowadays, Internet has become a primary technology in the human daily life. They use Internet not only to communicate each other but also to do business, militarism, research collaboration, public services, etc. It connects almost all places over the world. Internet users could be residential users, large business users and scientific users (O'Mahony, 2006). The explosive growth of Internet users caused Internet address depletion problem. Current Internet Protocol known as Internet Protocol version 4 (IPv4) has 2^{32} address spaces will be depleted on 2011 (Huston, G., 2007). To overcome the

problem IETF has developed a new Internet Protocol (Bradner, S., & Mankin, A., 1995) called IPv6. Compare with the former, this new protocol comes with big advantages including larger address space that is 2^{128} address spaces (Deering, S., Hinden, R., 1998), simpler header format (Hagen, S., 2006) and more extensibility (Davies, J., 2003). However, IPv6 packets transmission still uses the traditional TCP/IP protocol stack that has limitation on duplicate CRC calculation and regeneration in Data Link layer of every router (Braun, F., & Waldvogel, M., 2001). The limitation has caused IPv6 packets transmission cannot exploit the advantages of IPv6 features optimally.

In other hand, underlying technology both Data Link layer and Physical layer has grown very fast. Recently, for Data Link layer, we have gigabit Ethernet technology that is able to transmit large IPv6 packets on short time while Physical layer, we have fiber optic that has very low bit error rate (BER). The technologies support the high speed networks to get faster packets transmission. Transmission of IPv6 packets on this kind of technologies will reduce number of erroneous packet. As known, in the existing technology with fast Ethernet and copper as medium on IP packets transmission, the possibility of error is very low. Hence, using gigabit Ethernet as Data Link layer protocol and fiber optic as transmission medium will produce very low error possibility. Thus, duplicate CRC calculation and regeneration in every router will introduce high network latency and it is time consuming task.

Taking advantages of IPv6 features especially extension header extensibility as well as the advancement of high speed networks technology, we proposed CRC extension header (CEH) to do error detection in Network layer instead of in Data Link layer. CEH aims to eliminate the duplication of CRC code calculation and regeneration in every router. Thus, IPv6 packets transmission will be faster because of none of error detection processing in every router. The rest of this paper will discuss the related works on reducing duplicate CRC calculation and regeneration in Section 2. Theoretical consideration is in Section 3. Proposed mechanism and its experiment are explained in Section 4 and 5 respectively. The last section is conclusion of this paper.

2. Related Works

Problems in IPv6 packets transmission over high speed network are due to duplicate CRC verification and regeneration in every router along the network path. Many researchers have tried to find out solution of the problem. This section presents three of them that most related to this research. Braun and Waldvogel (2001) considered the IP packets processing in intermediate system that only few bits of the packets transmitted will change in each intermediate node. Most of the packet is kept to be constant especially the original data. In case of IPv4 packet, only 3 bytes which are TTL (time to live) field one byte and header checksum field two bytes will be changed in the forwarding node. In IPv6 packet, there is only a byte which is hop limit field is decreased by one after forwarding process. As only a small changing of the IP header in the intermediate node, it is no need to check overall packet for error detection. The authors proposed a method namely fast incremental CRC update. It distinguishes CRC computation of changing field and unchanged field. Sender generates both CRC code of two group's field and appends them to the packet. In intermediate node, CRC calculation is done only for the modified field. Thus, CRC computation becomes faster due to only few bits need to be calculated. Overall CRC computation of an IPv6 frame is only 15 bytes. This technique is effective to reduce CRC time processing in a router. However, it still needs to do the calculation in each intermediate node. Frequency of CRC calculation is still similar with the existing system. It uses CRC-32E as generator polynomial that has size of 32 bits to detect error in each router. CRC-32E was designed to calculate data length from 512 up to 1518 bytes. Thus, utilization of CRC-32 to compute only for 15 bytes in this technique is useless.

Weidong Lu and Stephan Wong (2004) also proposed a similar technique to reduce CRC calculation using fast CRC update. It implemented CRC parallel calculation. It separated CRC calculation of modified field and unmodified field of Data Link layer frame. In a router, calculation was only on modified field, and then combines the CRC code with original CRC code of non modified field. This is done in intermediate node instead of final receiver. This technique also still calculates

CRC in each node using CRC-32E. The difference with the previous method is location to do combination between CRC code of modified field and the one of non modified field of a frame.

The latest research was done by Satran et. al. (2005). The authors suggested out of order incremental CRC computation. The method utilized fragmentation process in IP packet transmission. Traditional protocol splits a message into the segments. Sender transmits each segment in different order and it may through different way. Usually, the receiver stores the segments till form the whole message. Then, it processes the message including CRC calculation. Accordingly, it needs more memory to store the segments, requires long latency and extra bandwidth. The method proposed to process each segment arrives to the receiver immediately without wait for entire message. After individual segment processing finished, it will deliver to upper layer directly. Thus, upper layer can process each segment arrive individually. It decreases latency and save memory without storing segments. The technique only reduces latency in destination and it does not touch intermediate node processing. Duplicate CRC calculation is in intermediate node. Hence, overall data transmission still needs more time to process the packets.

3. Theoretical Consideration

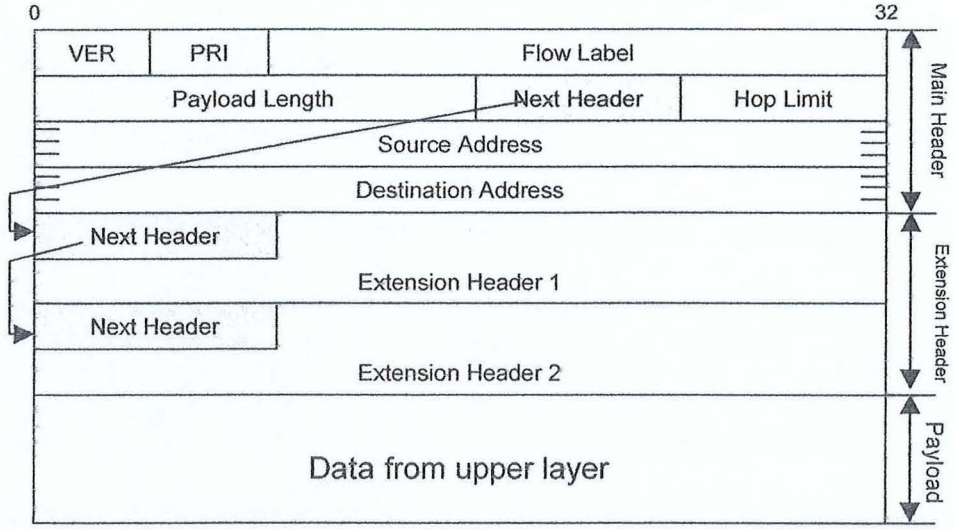
IPv6 is an enormous Internet technology that has been developing as an evolutionary of the existing Internet protocol, IPv4. It intended to improve quality of Internet services. As a future Internet technology, it has to meet the need of high speed data communication such as higher transfer rate, error free and real time application. To achieve the requirements, IPv6 was designed with some improvements of the former Internet protocol, IPv4. This section introduces one of the enhancements of the new Internet protocol technology which is IPv6 extension header.

3.1. The Concept of IPv6 Extension Header

An IPv6 packet comprises of header and upper layer payload. The IPv6 header consists of main header that has fixed size 40 bytes and extension header with optional size that is not limited to 40 bytes (Blancet, 2005). Figure 1 shows an IPv6 packet with extension header. The concept of IPv6 extension header was following option field in IPv4. According to (Deering, S., Hinden, R., 1998), concepts of IPv6 extension header are summarized as follow:

1. IPv6 extension header is optional and it is placed between main header and upper layer header in an IPv6 packet.
2. There are numbers of extension header in IPv6 including hop by hop options, destination option, routing header, fragmentation header, authentication header and encapsulating security payload. Each is identified by a distinct next header value determined by IANA.
3. An IPv6 packet may carry zero, one, or more extension headers and not limited to 40 bytes. Each is identified by the next header field of the preceding header or extension header.
4. With an exception of hop by hop options header, extension headers are not processed by any node along a packet's delivery path, until the packet reaches final destination node.
5. Extension headers must be processed strictly in the order they appear in the packet.

Figure 1: IPv6 Packet with Extension Header



There is no standard of uniform format for IPv6 extension header. RFC 2460 explained format of current extension header that has specific format depend on their function. Krisnan at al. (2008) proposed a uniform format of IPv6 extension header called GIEH (Generic IPv6 Extension header) as shown in Figure 2. It has four fields which are next header field, header extension length field, specific type field and header specific data field as explained the following.

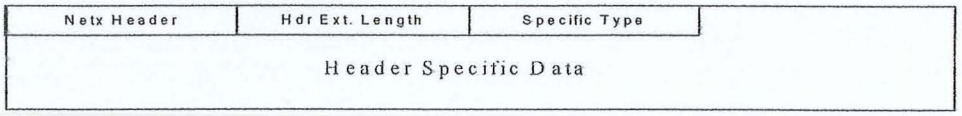
Next header: 8 bits selector to identify the type of Extension header immediately following this Extension header.

Hdr. Ext. Length: 8 bits unsigned integer that indicates the length of the Extension header in 32 bits units.

Specific Type: 8 bits unsigned integer that is the actual IPv6 Extension header type. This is allocated from IANA.

Header Specific Data: this is the core of Extension header that contains specific data as the requirement of the extension header. The length is variable and must be padded as needed in order to ensure that the whole extension header is a multiple of 8 bytes long.

Figure 2: Format of Generic IPv6 Extension Header Proposed by Krisnan

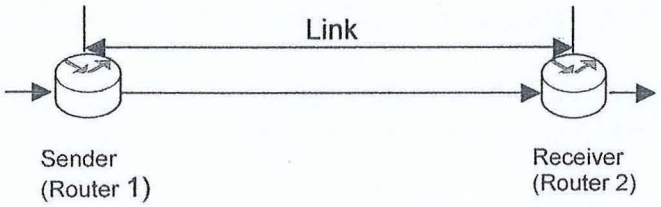


3.2. Data Link Layer Error Control

TCP/IP suite applies two stages of error control: lower layer error control to control transmission error and upper layer error control that cover higher layer level error. Problem of error control in this paper is about duplicate CRC calculation and regeneration in Data Link layer of every router. This type of error control that is done by frame check sequence (FCS) field is responsible to control error caused by transmission medium between two adjacent nodes. Figure 3 depicts connection of two nodes that apply link by link error control. In the mechanism, sender node (Router 1) buffers a copy of sending message until it receive acknowledgment from receiver node (Router 2). Receiver node generates ACK for each message received correctly. To determine whether the message valid or not, the receiver is equipped by

an error detection mechanism. Link by link error control usually implements cyclic redundancy check (CRC) to detect transmission error. The sender sets a specified time out period at the time it sends the frame. If an ACK is not received within the time out period, it is assumed the message is damage or lost. Then, sender retransmits the message.

Figure 3: Link by Link Connections



Router 1 receives frame from the preceding node, then it verifies the frame for error detection in Data Link layer. In case the frame is free from error, it passes to do Network layer processing otherwise discard the frame. Data Link layer of outgoing port of Router 1 gets the packet from Network layer and generates a new CRC code to do next hop error detection. Router 2 and the next router will do the same process as Router 1. This mechanism is time consuming task on the IPv6 packets transmission over high speed networks. This is because very rare transmission error caused by the medium (Dempsey, 1994). In addition, if we use fiber optic that has BER of 10^{-15}, it has very small possibility of transmission error.

Unfortunately, in a network with large intermediate system, it may have more than two links. Thus, total transmission time from sender to receiver that is the total transmission time of all adjacent nodes is very high. It means the network latency of packet transmission is also high. In the near future, it becomes a big problem on IPv6 packets transmission. Thus, decreasing of network latency is an important issue.

3.3. Cyclic Redundancy Check

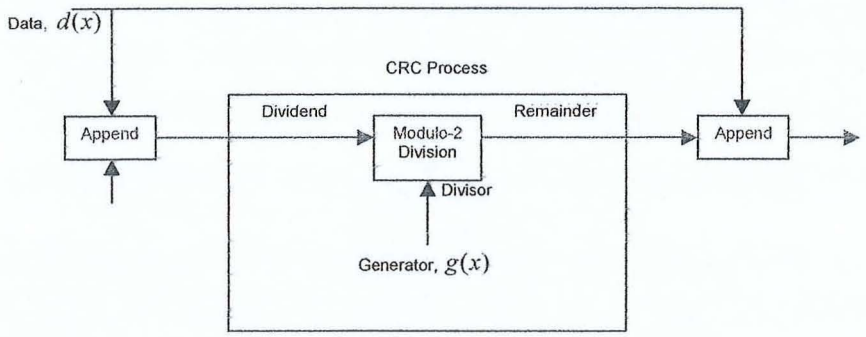
Cyclic redundancy check (CRC) is a technique to detect error in digital data transmission and storage system (Michael, E. K., & Frank, L. B., 2008). CRC is part of cyclic code because rotating of CRC code yields a new CRC code. It is also called linear code meaning addition one or more of CRC code will get a new CRC code. In digital data communication CRC is used to detect lower level error or error due to transmission medium. The number of error depends on the medium used to transmit a frame. Noisy channel will produce burst error while un-noisy channel has low probability of error in the form of independent (single) bit error. Capability of medium to produce error is known from its BER (bit error rate), low BER means low possibility of error occurred.

There are various algorithm of CRC calculation including algebraic approach, bit oriented approach and table driven approach (Stigge, M, at. al, 2006). The simplest way to explain CRC operation is algebraic approach. The operation is done by dividing the data word $d(x)$ by generator polynomial $g(x)$ using modulo-2 division.

$$d(x) = q(x).g(x) + s(x)$$

The division yields $q(x)$ as quotient of the division and $s(x)$ as the remainder. CRC operation intends to get $s(x)$ instead of $q(x)$. The remainder $s(x)$ is the CRC codes that will be transmitted with the packet. Before does the division, the data word need to append with 0s in the right. The 0s will be replaced by the obtained $s(x)$. This is done by multiplying the data word $d(x)$ with x^p. Schematic diagram of CRC code generation of a frame is shown in Figure 4.

Figure 4: CRC Code Generations

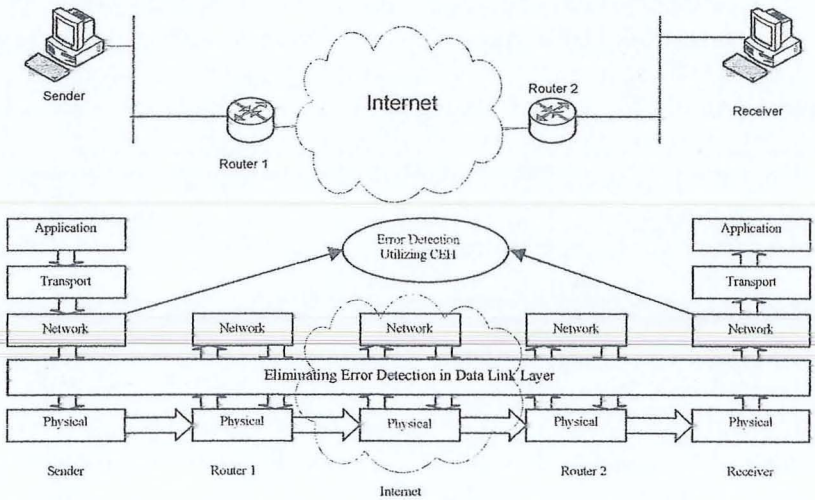


4. Proposed Error Detection Mechanism

Based on the theory of existing error control mechanism in Data Link layer that has a problem on duplicate CRC calculation and regeneration in every router, we propose CRC (cyclic redundancy check) Extension Header (CEH) to do error detection in Network layer. Utilizing IPv6 extension header as error detection will eliminate CRC calculation and regeneration in Data Link layer of every router. There is one generation process of CRC code only that is in sender machine and CRC code verification only which is in receiver machine. The proposed mechanism is shown in Figure 5.

In the proposed error detection mechanism, sender generates IPv6 packets in the Network layer and then generates CEH from the correspond packet excluding 1 byte hop limit field. The CEH is placed after destination address field and it is indicated by the next header field of IPv6 main header. The packet is delivered to Data Link layer to get Data Link header without trailer or frame check sequence (FCS). The packet is then sent through interconnecting devices (routers) to reach the receiver. All of the routers process the packet as usual excluding verification and regeneration of CRC code in Data Link layer. They also do not process the CEH inside IPv6 packet because CEH follows other general extension header that is just processed in destination node. The receiver receives the IPv6 packet transmitted and verifies the CEH code inside in its Network layer. When the receiver detects error in the packet received, it has to discard the packet and wait for retransmission.

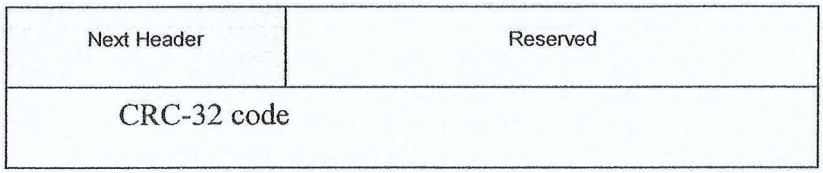
Figure 5: Proposed Error Detection Mechanisms



This idea is extremely different with the existing method that conducts error detection and correction in Data Link layer. This new mechanism does not require the Data Link layer to verify and regenerate CRC code for error detection in every router. Verification of CRC code will be done at Network layer of the destination node by processing CEH inside the IPv6 packet received. Routers just need to process IPv6 packet at their Network layer as usual which is simply a forwarding decision and hop limit updating. Theoretically, implementing this mechanism will reduce packet processing time in every router by deleting two processes of CRC code calculation. Thus, network latency of IPv6 packets transmission will also decrease significantly. In other word, IPv6 packets processing will be faster by utilizing the new protocol's features advantages.

The CEH has three fields: next header field, reserved field and CRC-32 code field as shown in Figure 6. Next header is an 8 bits indicator to point to other extension header following CEH such as TCP. The main field of CEH is CRC-32 code that generated from entire IPv6 packet excluding hop limit using modulo 2 division. Generation of CRC-32 code uses CRC-32C as generator polynomial (Supriyanto at al., 2009). It is not only cover the minimal size of IPv6 MTU 1280 bytes but also cover the future MTU such as jumbo frame.

Figure 6: Format of CRC Extension Header



The rest field of CEH format is reserved field that is allocated for future use instead of specific extension header type. The size of CEH follows the minimal size of IPv6 extension header which is 64 bits (Deering, S., Hinden, R., 1998). Thus reserved field has size of 24 bits. It can be used to enlarge the size of CRC code or other specific function.

5. Experiment

5.1. Experiment

To verify the acceptability of the new concept we did a experiment. The experiment has two scenarios: transmission of IPv6 packets with CEH as error detection in Network layer and transmission of IPv6 packets with FCS as error detection in Data Link layer. The first experiment simulates the new concept and the later simulates the existing system of error detection mechanism. The two experimental scenarios are expected to yield adequate data to justify performance of the new error detection proposed compared to the current error detection.

The topologies of the experiments follow mechanism in Figure 5 that has five nodes: sender, router 1, router 2 (Internet), router 3 and receiver. All of nodes are PC with Core 2 Duo processors that were configured as a mini IPv6-only network. Sender is an end system installed with IPv6 packets generator program that is able to generate various type of IPv6 packets required in the experiments including IPv6 packet with CEH and IPv6 packet without CEH. Router 1, 2 and 3 represent an intermediate system of the network. In the first experiment, they just forward the IPv6 packets with CEH rather than checking each packet for error. The intermediate system in the second experiment performs error detection in their Data Link layer of each router for each packet received before forwarding the packets to next path. The last node is the receiver which is a PC installed with the same program as the sender to verify all packets received especially CRC code inside the packet.

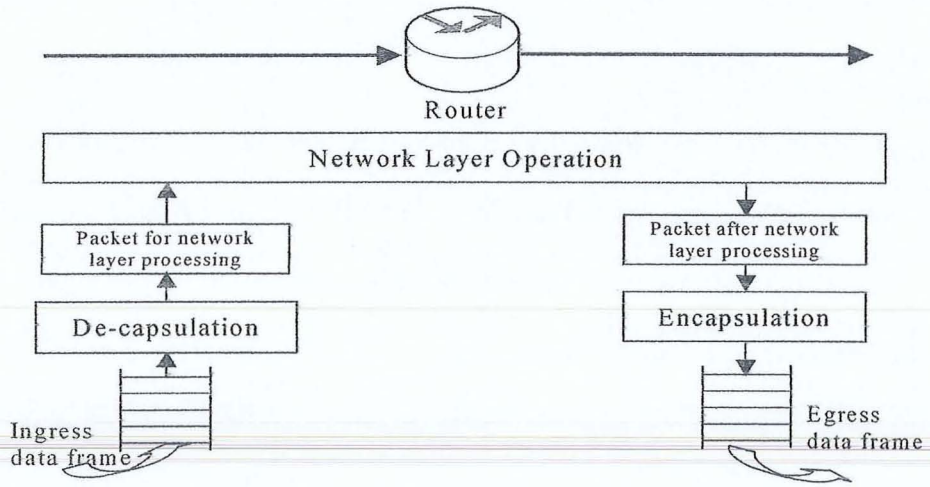
The sender generates IPv6 packet by adding IPv6 main header to a TCP segment received from Transport layer. The Network layer then generates CRC code from both IPv6 header and payload using algorithm in Figure 4. The CRC code obtained is inserted as CRC extension header into the IPv6 packet. The CEH is placed after destination address field before upper layer data. Data Link layer in the first scenario just adds the 14 bytes of link layer header. There is no trailer (FCS) in the Data Link layer frame that usually does the error detection task. The frame without frame check sequence (FCS) generated by the sender is then transmitted through the experiment network.

Because of no FCS inside the IPv6 frame, there is no longer need for each router to do CRC calculation and regeneration. The ingress data frame is just verified for the frame header by incoming port of router and passed to Network layer to determine the next path (forwarding process). The egress data frame also does not have Data Link layer trailer. Hence, IPv6 packets processing in every router is faster. Overall packets transmission also need less time compared to the existing system. The processing of IPv6 packet in a router is depicted in Figure 7.

The only node that will verify the CRC code inside IPv6 packet is final destination node (receiver) indicated by destination address field of the IPv6 packet. The verification is done to check whether the packet received is free from error or not. It is done by generate a new CRC code using algorithm in Figure 4. It compares the CRC code generated with the CRC code inside the IPv6 packet received. If the two CRC code is the same, there is no error inside the IPv6 packet received otherwise there is an error and discard the packet.

In the second experiment which is experiment of current error control mechanism, the sender generates IPv6 packet without any extension header. The packet is then encapsulated by Data Link layer by adding Data Link layer header and trailer. Thus we called the frame as IPv6 packet with FCS. Each intermediate node will receive the packet and process it as usual. They process the Data Link layer header including CRC code computation to detect transmission error for corresponding hop. The only packet that is justified as free from error packet will be delivered to Network layer to do forwarding process without processing any extension header. Before the IPv6 packet is forwarded to the next hop, the packet will be encapsulated again in Data Link layer of the outgoing port of the router including CRC code regeneration.

Figure 7: Packets Processing in a Router



Receiver node in the second experiment captures the IPv6 packet that addressed to it and the first packet processing is Data Link layer header and trailer verification including CRC code computation. Then, the correct packet will be passed to Network layer to do the next process which is IPv6 header verification. Otherwise, discard the erroneous frame and wait for retransmission. Receiver also notes the time needed to process the IPv6 packet in Data Link layer.

In the experiments, we use four metrics to measure performance of the proposed error detection mechanism those are processing time, delay and delay variation, packet error rate and packet loss rate. Processing time is time required to process an IPv6 packet in a node. It consists of processing time at sender, intermediate node and receiver. This metric is important to know the impact of implementing CEH as error detection in Network layer for network latency. Delay variation or jitter is comparison of two delay time of two sequence packets traveling from sender to receiver. As IPv6 packets travel to reach the destination, they pass through various network elements including routers. They may have different latency due to queue in a network element. Observation of delay variation will provide information about network queue and suitability of an application on the network.

Packet error occurs when the receiver detects an error within the packet received due to transmission medium. If the packet does not reach the destination is considered as packet loss. Packet loss and error rate is comparison between lost packet and erroneous packet with the amount of IPv6 packet sent by the sender. The last two metrics are important on studying the influence of inserting CEH on the IPv6 packet to the reliability of IPv6 packets transmission.

5.2. Experimental Result and Discussion

Based on the experiments, adequate results were obtained including processing time, network latency, delay variation and packet error. Total processing time resulted by the experiments is listed in Table 1. Table 1 is comparison of total processing time at sender and receiver between IPv6 packet transmission with CEH and IPv6 packet with FCS. The comparison shows processing time for IPv6 packet with CEH is higher of 15% average than IPv6 packet with FCS. This is due to processing of IPv6 packet with CEH is more complex than processing of IPv6 packet with FCS. It needs to separate hop limit field from the IPv6 packet first before calculate the CRC code and has to insert the CEH into the IPv6 packet after calculation. The IPv6 packets transmission with FCS do not need to do the two processes, instead, it just calculates CRC code and appends it in the frame. The Table also shows that short IPv6 packets sizes below 512 bytes have small differences (7%) and long packet size larger than 512 bytes have large differences (22%). As most Internet traffics are short packet, implementing of CEH on IPv6 packets transmission will be beneficial to the Internet community because it only adds small time to process CEH.

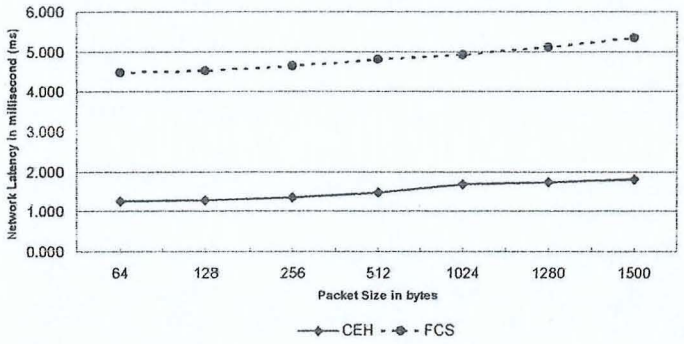
However, network latency of IPv6 packets transmission with CEH included all of processing time of node is lower of 68% average than FCS as shown in Figure 8. This is because the elimination of duplicate CRC calculation and regeneration in every router. Increasing of processing time in sender and receiver of 15% average for IPv6 packet with CEH is not significant compare to the decreasing of network latency of 68% average. It still can increase network performance for53 % in average. This percentage may be higher for a larger IPv6 network while processing time is constant when the machine is unchanged.

Table 1: Processing Time of CEH and FCS

Packet size (bytes)	64	128	256	512	1024	1280	1500
PT_{CEH} (millisecond)	1.257	1.285	1.355	1.483	1.681	1.730	1.804
PT_{FCS} (millisecond)	1.178	1.177	1.290	1.301	1.382	1.442	1.474
ΔPT (millisecond)	0.079	0.108	0.065	0.181	0.299	0.287	0.330
% ΔPT	7%	9%	5%	14%	22%	20%	22%

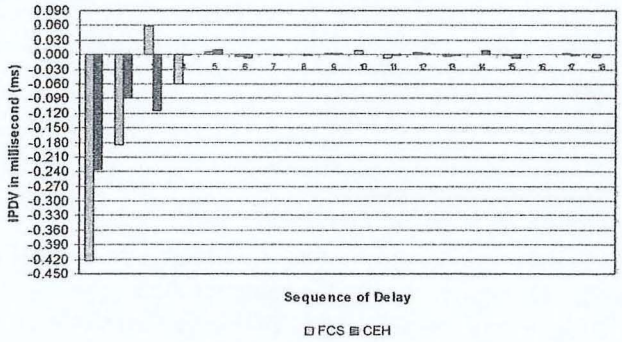
Both Table 1 and Figure 8 showed that processing time and network latency of IPv6 packet transmission using either CEH or FCS as error detection increases when packet size was increased. This is because processing of CRC calculation uses table lookup algorithm that process the packet byte by byte. Thus, bigger packet size need more time to do the processing.

Figure 8: Network Latency of CEH and FCS



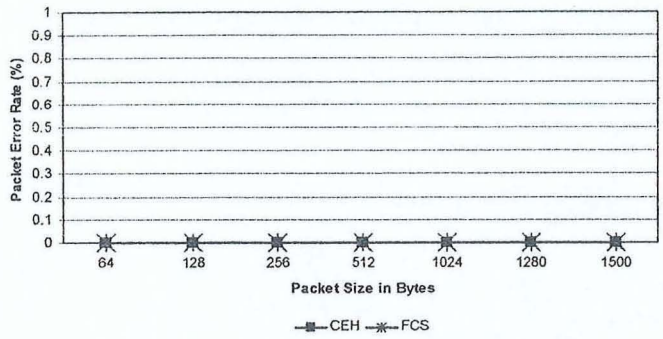
The experiments also resulted in the value of inter packet delay variation (IPDV) on transmission of IPv6 packet with CEH and transmission of IPv6 packet with FCS. Experiment result is shown in Figure 9. It exhibits graph of IPDV versus sequence of delay for the two types IPv6 packet transmission. The two graphs show transmission of IPv6 packet with CEH has smaller interval of IPDV than its competitor. This phenomenon indicates transmission of IPv6 packet with CEH as error control mechanism at Network layer has decreased the packet delay and buffer requirement in router. Eliminating error detection at Data Link layer decreased processing time of IPv6 packet in router. Thus, queuing problem can be decreased in every router.

Figure 9: Comparison of IPDV between CEH and FCS



Comparison of packet error rate for transmission of IPv6 packet with CEH and IPv6 packet with FCS is presented in Figure 10. Experiments were done in order to know the effect of utilizing CEH as error detection at Network layer on quality of IPv6 packet transmission in term of packet error and packet loss. The erroneous packets can be known from the comparison of CRC code of sending packet and receiving packet. If they are not equal, there is an error inside the packet. Figure 10 shows the percentage of IPv6 packet error rate on transmission using CEH as error detection at Network layer. Figure 10 exhibits there is no erroneous IPv6 packets on both transmission of IPv6 packet with CEH and IPv6 packet with FCS. It shows utilizing CEH as error detection mechanism at Network layer does not affect on IPv6 packet error rate.

Figure 10: Comparison of Packet Error Rate



Packet drop is an IPv6 packet that sent by the sender and does not reach the destination intended. It can be caused by transmission medium, congestion and long queue. As demonstrated in Table 1, processing time of IPv6 packets with CEH is high at sender and receiver. Thus, the specification of sender and receiver machine is important. High processing speed of machine processes the packet faster, otherwise the processing is slower. This is found in the experiments of measuring packet loss of IPv6 packet with CEH transmission. When Intel (R) Core™ 2 Duo processor was used, there is no packet loss in the transmission. In contrast, when Intel Pentium (R) D was used, there was packet drop in the transmission as shown in Table 2.

Table 2: Packet Drop on IPv6 Packets Transmission

Packet Size (bytes)	64	128	256	512	1024	1280	1500
Packet sent	145349	83333	44964	23408	11950	9600	8256
Packet Received[1]	145349	83333	44964	23408	11950	9600	8256
Packet Received[2]	145343	83333	44964	23407	11949	9600	8256
Packet Loss[1]	0	0	0	0	0	0	0
Packet Loss[2]	6	0	0	1	1	0	0

[1] is Intel (R) Core™ 2 Duo processor and [2] is Intel Pentium (R) D

6. Conclusion

In this paper we have proposed a new structure of IPv6 extension header called CRC Extension Header (CEH). The CEH was proposed to do error detection in Network layer on IPv6 packets transmission over high speed networks. The new concept eliminated error detection at Data Link layer. The CEH is generated by the sender machine and will be processed by the receiver machine. CEH comprises three fields: next header field, CRC-32 code field and reserved field. CRC-32 code is the main field that contains 4 bytes CRC code.

Elimination of error detection in Data Link layer of a router in the proposed method has decreased the network latency of IPv6 packets transmission. The result showed the network latency decrease by 68% in average compared to the normal latency. Unfortunately, the processing time of IPv6 packet with CEH is higher due to it has more complex processing than IPv6 packet with FCS. However, the value of processing time is very small compare to decreasing network latency on IPv6 packets transmission with CEH. The proposed error detection mechanism also showed good ability to detect transmission error inside the transmitted packet.

7. Acknowledgment

This work was supported in part by the USM-RU-PRGS under Grant 1001/PKOMP/832028 and also by the Ministry of National Education of the Republic of Indonesia.

References

[1] Blanchet, M. (2005). Migrating to IPv6: A Practical Guide to Implementing *IPv6 in Mobile and Fixed Networks*. Hoboken, N.J.: John Wiley & Sons.

[2] Bradner, S., & Mankin, A. (1995). *The Recommendation for the IP Next Generation Protocol.* RFC 1752. IETF.

[3] Braun, F., & Waldvogel, M. (2001). *Fast incremental CRC updates for IP over ATM networks.* Proceedings of IEEE Workshop on High Performance Switching and Routing, 2001.

[4] Davies, J. (2003). *Understanding IPv6*. Washington: Microsoft Press.

[5] Deering, S., Hinden, R. (1998). *Internet Protocol Version 6 (IPv6) Specification.* RFC 2460. IETF.

[6] Dempsey, B. J. (1994). Retransmission-Based Error Control for Continuous Media Traffic in Packet-Switched Networks. *PhD Dissertation*. University of Virginia

[7] Hagen, S. (2006). *IPv6 Essentials* (Second Edition). Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc.

[8] Huston, G. (2007). *IPv4 Address Depletion and Transition to IPv6*. The Internet Protocol Journal, Volume 10, Number 3.

[9] Michael, E. K., & Frank, L. B. (2008). *Novel Table Lookup-Based Algorithms for Hig-Performance CRC Generation.* IEEE Transaction on Computers. Vol. 57 No. 11. pp.1550-1560.

[10] O'Mahony, M. J., Politi, C., Klonidis, D., Nejabati, R. A. N. R., & Simeonidou, D. A. S. D. (2006). *Future Optical Networks*. Journal of Lightwave Technology, Vol. 24 No. 12, pp. 4684-4696.

[11] Satran, J., Sheinwald, D., & Shimony, I. (2005). *Out of Order Incremental CRC Computation.* IEEE Transactions on Computers, Vol. 54 No. 9, pp. 1178-1181.

[12] Stigge, M., Plotz, H., Muler, W., & Redlich, J.-P. (2006). *Reversing CRC Theory and Practice.* HU Berlin Public Report. http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2006-05/SAR-PR-2006-05_.pdf.

[13] Supriyanto, Abidah M. Taib, Rahmat Budiarto. *Selecting a Cyclic Redundancy Check (CRC) Generator Polynomial for CEH (CRC Extension Header)*, Proceedings of Internasional Conference on Quality in Research, Jakarta. 3 – 6 August 2009, ISSN 114-1284 pp. 245 – 250.

[14] Weidong Lu, & Wong, S. A *Fast CRC Update Implementation.* pp. 113-120. http://ce.et.tudelft.nl/publicationfiles/805_404_lu.pdf

FINANCIAL REPORT